On a unifying product form framework for redundancy models☆

U. Ayesta [a,b,c,e], T. Bodas [b,d,*], I.M. Verloop [a,d]

[a] CNRS, IRIT, 2 rue C. Camichel, 31071 Toulouse, France
[b] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[c] IKERBASQUE - Basque Foundation for Science, 48011 Bilbao, Spain
[d] Université de Toulouse, INP, 31071 Toulouse, France
[e] UPV/EHU, Univ. of the Basque Country, 20018 Donostia, Spain

Keywords:
Redundancy-d
Product-form distribution
Cancel-on-start
Independence assumption

A B S T R A C T

In this paper, we present a unifying analysis for redundancy systems with cancel-on-start (*c.o.s.*) and cancel-on-complete (*c.o.c.*) with exponentially distributed service requirements. With *c.o.s.* (*c.o.c.*) all redundant copies are removed as soon as one of the copies starts (completes) service. As a consequence, *c.o.s.* does not waste any computing resources, as opposed to *c.o.c.*

We show that the *c.o.s.* model is equivalent to a queueing system with multi-type jobs and servers, which was analyzed in Visschers et al., (2012), and show that *c.o.c.* (under the assumption of i.i.d. copies) can be analyzed by a generalization of Visschers et al., (2012) where state-dependent departure rates are permitted. This allows us to show that the stationary distribution for both the *c.o.c.* and *c.o.s.* models has a product form. We give a detailed first-time analysis for *c.o.s* and derive a closed form expression for important metrics like mean number of jobs in the system, and probability of waiting. We also note that the *c.o.s.* model is equivalent to Join-Shortest-Work queue with power of d (JSW(d)). In the latter, an incoming job is dispatched to the server with smallest workload among d randomly chosen ones. Thus, all our results apply mutatis-mutandis to JSW(d).

Comparing the performance of *c.o.s.* with that of *c.o.c.* with i.i.d. copies gives the unexpected conclusion (since *c.o.s.* does not waste any resources) that *c.o.s.* is worse in terms of mean number of jobs. As part of ancillary results, we illustrate that this is primarily due to the assumption of i.i.d. copies in case of *c.o.c.* (together with exponentially distributed requirements) and that such assumptions might lead to conclusions that are qualitatively different from that observed in practice.

1. Introduction

Using redundancy to minimize latency in parallel server systems has become very popular in recent years [1–6]. While there are several variants of a redundancy-based system, the general notion of redundancy is to create multiple copies of the same job that will be sent to a subset of servers. By allowing for redundant copies, the aim is to minimize the system latency by exploiting the variability in the queue lengths of the different queues. Several recent works have both empirically [1,2,7,8] and theoretically [3–5,9–11] indicated that redundancy can help in reducing the response time of a system.

☆ Research partially supported by the French "Agence Nationale de la Recherche (ANR)" through the project ANR-15-CE25-0004 (ANR JCJC RACON).
* Corresponding author at: CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France.
E-mail address: tejaspbodas@gmail.com (T. Bodas).

https://doi.org/10.1016/j.peva.2018.09.008

Broadly speaking, depending on when replicas are deleted, there are two classes of redundancy systems: cancel-on-start ($c.o.s$) and cancel-on-completion ($c.o.c$). In redundancy systems with $c.o.c$, once one of the copies has completed service, the other copies are deleted and the job is said to have received service. On the other hand, in redundancy systems with $c.o.s$, copies are deleted as soon as one copy starts being served. From a practical point of view, if servers have similar computing capacity the $c.o.s$ system is preferable: both configurations require the same amount of signaling among servers, but the $c.o.s$ system does not waste any computation resources. Having said that, most of the recent literature has focused on systems with $c.o.c$, and $c.o.s$. has largely remained elusive to exact analysis.

In a recent series of papers on the $c.o.c$. model, Gardner et al. [5,6] have provided a thorough analysis of the $c.o.c$ based redundancy system with K servers each with their own queue. In the redundancy-d model of [6], redundant copies of an arriving job are sent to $d \leq K$ homogeneous servers chosen uniformly at random. Under the additional assumptions that service times are exponentially distributed and that the redundant copies are independent and identically distributed (we call this the *independence assumption*), Gardner et al. have shown that the steady-state distribution has a product form. Further, they derive the mean response time in closed form and show that the stability region, i.e., the arrival rate for which steady state exists, does not reduce when compared to a system with no redundancy ($d = 1$). The latter seems counter-intuitive, because having multiple copies of the same job should imply more work for the servers and as a consequence result in a reduction of the stability region. The reason that this does not happen is due to the *independence assumption* made in [6]. The exponential assumption together with i.i.d. copies creates a situation where the effective departure rate of a job is d times faster than that in a single server. One of the objectives in this paper is to further assess the impact of this assumption on the performance.

In this paper, we provide to the best of our knowledge, the first analysis of a redundancy-d system with $c.o.s.$ for $1 \leq d \leq K$. We assume exponentially distributed service times, but copies do not need to be i.i.d. This is because for the $c.o.s.$ model, whether the copies are i.i.d. or not is irrelevant for the analysis and hence we do not require the *independence assumption*. We adopt as a benchmark the same underlying multi-server topology as the one of [6]. This model is very convenient, everything is symmetric and homogeneous and this permits us to isolate the impact of the redundancy scheme. We first show that $c.o.s$ is equivalent to a system with a single central queue and multi-type jobs and multiple servers that was analyzed by Visschers et al. [12]. This allows us to conclude that the steady-state distribution of $c.o.s$ is of product-form. In addition, we obtain for the first time an expression for the generating function for the number of waiting jobs and the mean number of jobs in the redundancy-d system with $c.o.s$.

We then show that redundancy-d with $c.o.s$ is equivalent to Join-Shortest-Work queue with power of d (JSW(d)). In the latter, an incoming job is dispatched to the server with smallest workload among d randomly chosen ones. The redundancy model hence represents a method of implementing JSW(d) without requiring to know the residual work in each of the queues. In addition, performance measures obtained for the $c.o.s$ model carry over to JSW(d).

Finally we extend and adapt the model of Visschers et al. [12] in such a way that by using a new state descriptor (that is different from [12]), we are able to analyze the redundancy system with $c.o.c$ and i.i.d. copies. We believe that our results open an important avenue for future work by establishing the link between redundancy models and the central queue model of [12]. A summary of our main contribution is given below:

1. We obtain a unifying approach to derive a product form stationary distribution for both $c.o.s.$ and $c.o.c.$ models (Propositions 1 and 7).
2. We provide the first exact analysis for the redundancy-d system with $c.o.s.$ and derive the generating function for the number of waiting jobs and its mean (Proposition 2). A key interpretation from the generating function is the fact that the number of waiting jobs is a mixture of sums of geometric random variables.
3. By allowing redundant copies that cancel-on start, we achieve the performance of JSW(d) without requiring to know the job sizes and the workload in the servers (Proposition 3). Thereby we also provide the first exact analysis for the performance under JSW(d) (Corollary 1).
4. In Proposition 6, we generalize the modeling framework of [12] and use this generalization to provide a novel method to analyze the stationary distribution for redundancy-d systems with $c.o.c$.
5. Numerically we observe that the impact of the modeling assumption of independent copies is non-negligible.

The rest of the paper is organized as follows. In the next section, we look at some of the related work and discuss the model and preliminaries in Section 3. We analyze the $c.o.s.$ model in Section 4, its equivalence to JSW(d) in Section 5 and discuss some asymptotic regimes in Section 6. We give an alternative product form for the $c.o.c.$ model in Section 7 followed by some numerical results in Section 8.

2. Related work

The main motivation to investigate redundancy models comes from empirical evidence suggesting that redundancy can help improve the performance in real-world applications. For example Vulimiri et al. [2] illustrate the advantages of redundancy in a DNS query network where a host computer can query multiple DNS servers simultaneously to resolve a name. Deal et al. [8] note that Google's big table services use redundancy in order to improve latency.

Most of the literature deals with the $c.o.c$ model with exponentially distributed service times and the *independence assumption*. Gardner et al. [5,6] provide a comprehensive analysis of redundancy $c.o.c$ queueing models. In [5] the authors

consider a class-based model where redundant copies of an arriving job type are dispatched to a type-specific subset of servers, and show that the steady state distribution has a product form. In [6], the previous result is applied to analyze a multi-server model with homogeneous servers where incoming jobs are dispatched to randomly selected d servers. Bonald et al. [13,14] have shown that the $c.o.c.$ model under the *independence assumption* and the balance fairness allocation yields the same steady-state distribution. An important result for the $c.o.c.$ model with *independence assumption* is that the necessary stability condition that the arrival rate per server is smaller than the service rate, is also sufficient, see both [13] and [6]. Another important observation made in [13] is that the $c.o.c.$ model is a special case of the Order Independent queue [15], which provides a direct method to derive the steady-state distribution. However we note that the departure rates from a state under the $c.o.s.$ model fail to satisfy the order independence condition, which in turn implies the necessity of a different approach from that used in [13] to analyze $c.o.s.$ We give more details in Section 7. In a recent paper [16], Adan et al. show the equivalence of the c.o.c. model with two other parallel service models.

A more general redundancy model is the so-called (n, k, r) model proposed and studied by Joshi et al. [17]. In this model there are n servers, r redundant copies and a job is said to be served if k copies are served. For this model, in [17], the authors analyze the impact of the service time distribution on the performance. Shah et al. [4] have showed that under i.i.d. memoryless service the average response time reduces as r increases, and that $r = n$ minimizes the average latency among all possible redundancy policies.

To the best of our knowledge, the only results available for the $c.o.s.$ (and $JSW(d)$) model have appeared in [17, Section 4], [11, Section 9] and [18]. In the former two, the analysis is restricted to the particular case in which redundancy copies are sent to all the servers, and in the latter, $JSW(d)$ is analyzed in the mean-field regime.

Focusing on the *independence assumption*, Gardner et al. [11] propose a more accurate model in which the service time of a redundant copy is decoupled into two components, one related to the inherent job size of the task, and the other related to the server's slowdown. As written in the abstract of [11] "The independent runtime model is unrealistic, and has led to theoretical results which can be at odds with computer systems implementation results". One of our contributions is to provide some numerical examples of this statement, and to provide an exact analysis for a redundancy model ($c.o.s.$) that does not need the *independence assumption*.

Redundancy models have also been studied in other application domains. For example an analysis of redundant models in the context of storage systems is provided in [9], and Ananthanarayanan et al. [1,7] have studied the benefits of redundancy in straggler mitigation in data centers and cloud computing.

3. Model, preliminaries and notation

In this section, we will first describe the redundancy-d model with $c.o.s.$ and $c.o.c.$, and discuss the impact of the independence assumption by looking at the case $d = K$. We then give some common preliminaries for the two redundancy models which is followed by a brief description of the multi-type job and server model of Visschers et al. [12]. This model forms the basis for our analysis of the two redundancy models.

3.1. Redundancy-d with c.o.s. and c.o.c.

We consider a generic redundancy-d model of K homogeneous servers each with a first-in first-out (FIFO) queue. The service rate of each server is denoted by μ. Jobs arrive according to a Poisson process with rate λ and have an exponentially distributed service requirement with unit mean. An arriving job chooses d out of K servers uniformly at random and sends d copies of the same job to these *feasible* servers. For a redundancy-d model, we will say that all jobs that choose the same d servers are of the same type. In all, there are $\binom{K}{d}$ job types and the arrival rate of any job type is $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$.

Under *cancel-on-start (c.o.s.)*, once any of the copies is taken for service, the remaining copies are canceled immediately. Further, on arrival of a job, if more than one of its d *feasible* servers are idle, then the job is served immediately at one of the idle *feasible* servers based on a predefined *assignment rule* and the remaining copies are canceled. For this $c.o.s.$ model, a natural *assignment rule* is the *uniform assignment rule* that chooses an idle *feasible* server uniformly at random.

Under *cancel-on-complete (c.o.c.)*, when one of the d copies of a job completes its service, the other copies are removed. In addition, for the $c.o.c.$ model we assume that the service requirements of the copies of the same job are independent. We refer to this assumption as *the independence assumption*. As stated earlier, *the independence assumption* was required for the product form analysis of [6] and such assumption is however not required for the analysis of the $c.o.s.$ model.

Before giving the preliminaries, we first illustrate the impact of the independence assumption for the particular case of $d = K$.

Example 1 (*Case $d = K$: Impact of Independence Assumption*)**.** When redundant copies are independent, the $c.o.c.$ model with $d = K$ is equivalent to a single-server queue with arrival rate λ and service rate μK.[1] This implies that the mean number of jobs in the system is $\rho/(1-\rho)$ where $\rho := \frac{\lambda}{K\mu}$. If we were to drop the *independence assumption* and instead assume that all copies have *exactly the same service requirement*, then the $c.o.c.$ model with $d = K$ reduces to K completely synchronized

[1] This can be seen as follows: starting from an empty queue, a first job will have copies in all K servers. A copy will hence finish at rate μK, after which a new job starts being served in all servers.

single-server queues, each having an arrival rate of λ and a service rate of μ.[2] The stability condition is now $\lambda < \mu$, i.e., $\rho < \frac{1}{K}$ which is considerably smaller than the stability condition under the independence assumption ($\rho < 1$). In addition, without the independence assumption, the mean number of jobs increases significantly to $\rho K/(1 - K\rho)$.

Turning now to the *c.o.s.* model, when $d = K$, with or without the independence assumption, this model is equivalent to an $M/M/K$ queue (see Appendix E for details). Clearly, while the *c.o.c.* model with *independence assumption* significantly outperforms the *c.o.s.* model, the performance of *c.o.c.* degrades severely when the *independence assumption* is dropped. Further, the stability region is also reduced illustrating the unfair advantage due to the *independence assumption*.

Recall that the main motivation behind the use of redundancy models was to improve the delay performance by sending copies to d servers simultaneously. However, in case of *c.o.c.* without the *independence assumption*, implementing full redundancy ($d = K$) results in a much *worse* performance than when there is no redundancy ($d = 1$). This is because $d = 1$ corresponds to Bernoulli routing in which case we have K independent $M/M/1$ queues with rates λ/K and μ which has a better performance as compared to K synchronized single servers with rates λ and μ (*c.o.c.* with $d = K$ without the *independence assumption*). Under *c.o.s.*, such an inefficiency does not arise thereby emphasizing on the importance of *c.o.s.* for redundancy systems with identical copies.

3.2. Preliminaries

3.2.1. Stability condition

We define the total load as $\rho = \frac{\lambda}{K\mu}$ and assume $\rho < 1$. For both the *c.o.s.* and *c.o.c.* models, $\rho < 1$ characterizes the stability condition under which a stationary distribution exists. This was proved in [5] for *c.o.c.* For *c.o.s.* it follows from the symmetry in the model where each server serves a fraction $1/K$ of the jobs. Since the service rate of a server is μ, each server is stable if $\lambda/K < \mu$, i.e., $\rho < 1$.

Mean number of busy servers: Let $p(i)$ denote the probability that there are i servers busy. Using the rate balance principle, we have (for both *c.o.s.* and *c.o.c.*)

$$\lambda = \mu \sum_{i=0}^{K} ip(i). \tag{1}$$

Hence, the mean number of busy servers in a redundancy-d system is $\sum_{i=0}^{K} ip(i) = \frac{\lambda}{\mu} = \rho K$.

3.2.2. Probability of an idle server

We now identify the probability of an arbitrary server (say server 1) being idle. Let P_1 denote the probability that server 1 is idle. Given i busy servers, let q_i denote the probability that server 1 is idle. We have,

$$\begin{aligned} P_1 &= \sum_{i=0}^{K} p(i)q_i \\ &= \sum_{i=0}^{K} p(i)\frac{\binom{K-1}{i}}{\binom{K}{i}} \\ &= 1 - \rho, \quad \text{from Eq. (1).} \end{aligned}$$

3.2.3. Central queue architecture

An important feature of both the redundancy-d models is that they can be represented by a *central queue architecture* with multiple servers. This fact was particularly useful in obtaining the stationary distribution for the *c.o.c.* model (see Gardner et al. [5]).

While a job in the traditional parallel queue representation is represented by its d copies (a copy in each queue of its d feasible servers), the same job is represented only once in the central queue (albeit by a type label to indicate its d *feasible* server). See Fig. 1 for the representation of a central queue in terms of *c.o.c.* and *c.o.s.* models. The jobs in the *central queue architecture* are ordered in a FIFO manner based on the knowledge of the sequence of job arrivals. A key interpretation with the *central queue architecture* is that when a server is free, it scans the central queue in FIFO order and (after skipping over infeasible jobs) chooses the oldest *feasible job* for service. For the *c.o.c.* model, multiple feasible servers can serve a job in the central queue. In the traditional parallel queue representation, this corresponds to the case when redundant copies of the job are in service in different queues. For example in Fig. 1, for the *c.o.c.* model, job (1, 2) in the central queue is being served by servers 1 and 2 simultaneously. In the traditional representation for the *c.o.c.* on the left, copies of this job are in service in queues 1 and 2 respectively. On the contrary, due to the cancel-on-start feature, we require that each job in the *central queue architecture* for the *c.o.s.* model is served by only one server. Further, if multiple idle feasible servers can serve an arriving job, then ties are broken using the *uniform assignment rule*.

[2] A job will be served simultaneously in all K servers. Since the service requirement is identical, all copies finish after an exponential time with mean μ.

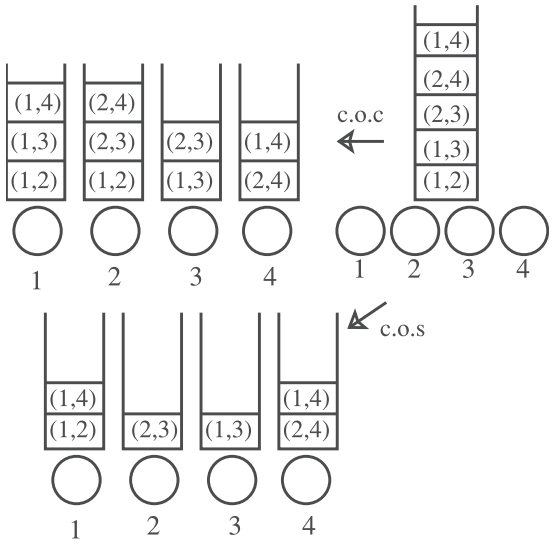

Fig. 1. A central queue representation for redundancy-2 $c.o.s.$ and $c.o.c.$ In the central queue, jobs are denoted by their feasible servers and are arranged in a FIFO order. Idle servers scan the central queue and serve the oldest feasible job. While multiple servers can serve a job in $c.o.c.$, the choice of server in $c.o.s.$ is by the *uniform assignment rule*.

Given the *central queue architecture*, note that the $c.o.s.$ and $c.o.c.$ models differ in their departure rate for jobs in service. Yet, it is this architecture that enables us to offer a unifying framework to analyze the two models. This unifying analysis is based on the multi-type job and server model of Visschers et al. [12] which also has a *central queue architecture*. For $c.o.s.$, we note that the system model (and hence the departure rate of jobs in the central queue) coincides with the model of [12]. This enables us to use [12] to analyze redundancy-d with $c.o.s.$ in detail in Section 4. For the $c.o.c.$ model, since multiple servers can serve a feasible job in the central queue, a direct application of Visschers et al. [12] is not possible. Additionally, the departure rates from jobs in the central queue of a $c.o.c.$ system depend on jobs that are present ahead in the central queue. In Section 7, we give a novel analysis for redundancy-d with $c.o.c.$ model that is based on an extension, together with an adaptation of the state descriptor of Visschers et al. [12]. As a first step, we provide an extension of Visschers et al. [12] in Section 7.1 to allow state dependent departure rates from the central queue (as required for the $c.o.c.$ model). We then adapt its state descriptor to $c.o.c.$ by introducing the notion of a super-server that is unique per job type (Section 7.2). The super-servers with state dependent departure rates now mimic the central queue of a $c.o.c.$ model leading to a novel analysis of its stationary distribution. Thus, our extension of [12] provides a comprehensive analysis for redundancy systems with $c.o.s.$ and $c.o.c.$

3.3. Multi-type job and server model of [12]

In this section we present the multi-type job and multi-type server model and the results as presented in Visschers et al. [12]. Visschers et al. [12] consider a *central queue* system with K possibly heterogeneous servers and multi-type jobs. Let $\mathcal{C}$ denote the set of job types and $\mathcal{M} = \{m_1, \ldots, m_K\}$ denote the set of servers respectively. For $a \in \mathcal{C}$, S_a denotes the set of *feasible* servers that can serve jobs of type a. Similarly, associated with a server $M \in \mathcal{M}$ is a set of feasible job types it can serve, denoted henceforth by $C(M)$. The jobs of type $a \in \mathcal{C}$ arrive according to a Poisson process with rate λ_a and the total arrival rate $\lambda = \sum_{a\in\mathcal{C}} \lambda_a$. All jobs have an exponential service requirement with unit mean and the service rate of server $m_i \in \mathcal{M}$ is denoted by μ_i for $1 \leq i \leq K$. Each arriving job waits in a *central queue* and is served in a first-in first-out (FIFO) basis. Each job can only be served by one of its *feasible* servers.Further, when a server becomes idle, it scans the central queue (in FIFO order) and picks the first feasible job that it finds for service. If an arriving job finds that more than one of its *feasible* servers are idle, then an *assignment rule* decides which of the idle *feasible* server is to be assigned to the arriving job.

State space representation

The multi-type job and server model was analyzed in [12] using a Markovian descriptor of the type $(n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$. This Markovian descriptor characterizes states of the system with i busy servers that are denoted by $M_1, \ldots, M_i$ and n_j waiting jobs between servers M_j and M_{j+1} for $1 \leq j \leq i-1$. The fact that this descriptor is Markovian can be seen from the state transitions and the corresponding transition rates from any generic state $s = (n_i, M_i, \ldots, n_1, M_1)$. For sake of completeness, we outline these state transitions and the transition rates in Appendix F. (See [12] for exact details.) In the rest of this section, we describe the meaning of this descriptor and introduce the notation for the transition rates that are required for the state transitions in Appendix F. We end this section by recalling the main result from [12] which says that this descriptor has a product form stationary distribution.

We will denote the state space for the multi-type job and server model by $\mathcal{S}$ where any generic state $s \in \mathcal{S}$ is of the type $s = (n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$. In this state space representation, jobs and active servers are arranged in a FIFO basis from right to left. Therefore all n_j jobs have arrived before n_k jobs where $1 \leq j < k \leq i$. Note that a job is waiting in the central queue only if all of its *feasible* servers are busy (serving jobs that came before it). Therefore n_1 denotes the number of those jobs (who have arrived before n_j jobs where $j > 1$) that have to wait since they have server M_1 as their only *feasible* server which happens to be busy. Let us denote the set of such job types by $\mathcal{U}(\{M_1\})$ where clearly $\mathcal{U}(\{M_1\}) = \{a \in \mathcal{C} : S_a = \{M_1\}\}$. Similarly, n_j represent jobs (that arrived after n_{j-1} but before n_{j+1} jobs) that have to wait because their feasible servers are busy. The feasible servers for these n_j jobs must clearly be a subset of the active servers $\{M_1, M_2, \ldots, M_j\}$ ahead of it (otherwise the job would not have been waiting if any of its feasible server was idle). Therefore for $1 \leq j \leq i$, we have $\mathcal{U}(\{M_1, M_2, \ldots, M_j\}) := \{a \in \mathcal{C} : S_a \subseteq \{M_1, \ldots, M_j\}\}$ as the set of possible job types for n_j waiting jobs in the representation.

To define the transition rates, let $\lambda_M(\{M_1, \ldots, M_i\})$ denote the activation rate of server M, when the set of active servers in state s is given by $\{M_1, \ldots, M_i\}$. In state s, this is the arrival rate of those customers that will make an idle server M to be busy. Note that this is also the transition rate from state s to state $(0, M, s)$. This activation rate $\lambda_M(\{M_1, \ldots, M_i\})$ depends on the *assignment rule* defined for the model, which determines the choice of an idle feasible machine (if any) for a job to be served. Next define $\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}$ as the arrival rate of jobs whose feasible servers are a subset of $\{M_1, M_2, \ldots, M_j\}$. We have

$$\lambda_{\mathcal{U}(\{M_1,M_2,\ldots,M_j\})} = \sum_{a \in \mathcal{U}(\{M_1,M_2,\ldots,M_j\})} \lambda_a. \tag{2}$$

In state $s = (n_i, M_i, \ldots, n_1, M_1)$, $\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}$ denotes the arrival rate of customers that do not find any feasible server to be idle. This is also the transition rate from s to $(n_i + 1, M_i, \ldots, n_1, M_1)$.

To specify the departure rates, note that for any state $s \in \mathcal{S}$, an active server $M_j \in \mathcal{M}$ has a fixed service rate μ_{M_j}. For this model, we define a *departure set function* $\mu_{\{\cdot\}}$ ($\{\cdot\}$ is used to denote a set) which is given by

$$\mu_{\mathcal{Y}} = \sum_{t \in \mathcal{Y}} \mu_t \text{ for } \mathcal{Y} \subseteq \mathcal{M}. \tag{3}$$

In words, the *departure set function* $\mu_{\mathcal{Y}}$ quantifies the total departure rate from a state where the set of active servers is $\mathcal{Y}$. In state $s = (n_i, M_i, \ldots, n_1, M_1)$, the total departure rate is given by $\mu_{\{M_1 \ldots M_i\}} = \sum_{j=1}^{i} \mu_{M_j}$. For sake of convenience, define $\hat{\mu}_{M_j}(s)$ to be departure rate in state s from any server M_j with $1 \leq j \leq i$. It is easy to see that

$$\hat{\mu}_{M_j}(s) = \mu_{\{M_1 \ldots M_j\}} - \mu_{\{M_1 \ldots M_{j-1}\}} = \mu_{M_j}. \tag{4}$$

The notation $\hat{\mu}_{M_j}(s)$ will be useful in Section 7.1 where we consider state dependent departure rates from servers.

The quantities $\lambda_M(\{M_1, \ldots, M_i\})$, $\lambda_{\mathcal{U}(\{M_1,M_2,\ldots,M_j\})}$, $\mu_{\{\cdot\}}$ and $\hat{\mu}_{M_j}(s)$ are now sufficient to describe the transition rates and state transitions for the model. We do so in Appendix F which also illustrates the Markovian nature of the state space representation.

Product form stationary distribution

By considering *assignment rules* that satisfy the following *assignment condition*, Visschers et al. [12] were successful in obtaining a product form stationary distribution.

Assignment Condition. *For $i = 1 \ldots K$, and for every subset $\{M_i, \ldots, M_1\}$ of $\mathcal{M}$ of size i, the following holds for every permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots, M_i$:*

$$\prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots, M_{j-1}\}) = \prod_{j=1}^{i} \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\}). \tag{5}$$

We now recall the following theorem from [12] that provides the product form stationary distribution.

Theorem 1. *For the multi-type job and server model with an assignment rule satisfying the assignment condition, the steady state probability for all states $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$, is given by*

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_1}^{n_1} \frac{\Pi_\lambda(\{M_1, \ldots, M_i\})}{\Pi_\mu(M_i, \ldots, M_1)} \pi(0) \tag{6}$$

$$\text{where} \quad \Pi_\lambda(\{M_1, \ldots, M_i\}) = \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}),$$

$$\Pi_\mu(M_i, \ldots, M_1) = \prod_{j=1}^{i} \mu_{\{M_1, \ldots M_j\}},$$

$$\alpha_j = \frac{\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}}{\mu_{\{M_1,\ldots M_j\}}} \text{ and}$$

$$\mu_{\{M_1\ldots M_i\}} = \sum_{j=1}^{i} \mu_{M_j}.$$

4. Redundancy-d with $c.o.s.$: An exact analysis

In this section, we provide the first exact analysis for the redundancy-d model with $c.o.s.$ To recall some of the preliminaries on the redundancy-d model with $c.o.s.$, multi-type jobs ($\binom{K}{d}$ in number) arrive according to a Poisson process with an arrival rate of $\frac{\lambda}{\binom{K}{d}}$ per type. The type of an arriving job is determined by its choice of d *feasible* servers where copies of the job are replicated. When any one of these copies is taken for service, the remaining copies are deleted. Additionally, on arrival of a job, if multiple *feasible* servers are idle, then one of these idle feasible servers is chosen uniformly at random to serve the job (*uniform assignment rule*). We also noted that the redundancy-d model with $c.o.s.$ can alternatively be represented by a *central queue architecture* where multi-type jobs wait in a central queue in a FIFO manner and where idle servers scan the central queue in search of feasible jobs (see Fig. 1). But the redundancy-d model with $c.o.s.$ as described above is also exactly the system setting for the multi-type job and multi-server model (Visschers et al. [12]) that was described in the previous section. This observation will now allow us to analyze the $c.o.s.$ model using Theorem 1.

For the $c.o.s.$ model, we will use the Markovian descriptor $(n_i, M_i, n_{i-1}, M_{i-1}, \ldots, n_1, M_1)$ to represent any state $s \in \mathcal{S}$ (following [12], see Section 3.3). Recall that in this representation, active servers and waiting jobs are arranged in a FIFO basis from right to left. For a given state s, n_j $(j \geq d)$ denotes the number of waiting jobs that have their d feasible servers to be a subset of $\{M_j, \ldots, M_1\}$ and have arrived after the first $n_1 + \cdots + n_{j-1}$ waiting jobs. In fact, recall from the discussion in Section 3.3 that the type of these n_j waiting jobs belongs to the set $\mathcal{U}(\{M_1, M_2, \ldots, M_j\})$. A key observation to be made is that since each job type has d feasible servers, it can never happen that there are jobs in the central queue waiting to be served and that there are less than d busy servers in the system. Therefore, $n_1 = \cdots = n_{d-1} = 0$. For the redundancy-2 $c.o.s.$ example of Fig. 1, using the above description it can be seen that the state space for the example can be represented as $(1, M_4, M_2, M_3, M_1)$ where M_i represents the ith server and the number 1 at the end indicates the waiting customer $(1, 4)$.

Recall that to apply Theorem 1, we need to consider an *assignment rule* that satisfies the *assignment condition*. We consider the *uniform assignment rule*: if an arriving job finds α of its d $(\alpha \leq d)$ servers busy, then the probability of choosing each idle server for service is $\frac{1}{d-\alpha}$. For the *uniform assignment rule*, we prove that it satisfies the *assignment condition*. The proof is in the Appendix.

Lemma 1. *The uniform assignment rule satisfies the assignment condition given by Eq. (5).*

4.1. Product form stationary distribution

In this section we provide the product form steady state distribution for redundancy-d system with $c.o.s.$ for the Markovian descriptor $s = (n_i, M_i, \ldots, n_1, M_1)$ by applying Theorem 1 to the $c.o.s.$ model. (See the Appendix for the proof.)

Proposition 1. *The steady state distribution for any state $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ is given by*

$$\pi(s) = \begin{cases} G^i(K,d)\frac{\pi(0)}{i!\mu^i} & \text{for } i < d \text{ and } n_1 = \cdots = n_{d-1} = 0, \\ {r_i}^{n_i} \ldots {r_d}^{n_d} G^i(K,d)\frac{\pi(0)}{i!\mu^i} & \text{for } i \geq d \text{ and } n_1 = \cdots = n_{d-1} = 0, \\ 0 & \text{elsewhere} \end{cases} \tag{7}$$

where $\pi(0)$ is the probability of an empty system and where

$$G^i(K,d) = \prod_{j=1}^{i} G_j(K,d),$$

$$G_j(K,d) = \frac{\lambda}{\binom{K}{d}} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha} \text{ and}$$

$$r_i = \frac{\lambda\binom{i}{d}}{i\mu\binom{K}{d}}. \quad \square$$

The stationary distribution for any state s can be written in an alternative form as follows. Define

$$\bar{G}_j(K,d) = \frac{G_j(K,d)\cdot K}{\lambda} = \frac{d}{\binom{K-1}{d-1}} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha},$$

$\bar{G}^i(K,d) = \prod_{j=1}^{i} \bar{G}_j(K,d)$ and $\bar{r}_i = \frac{r_i}{\rho} = \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}}$. The stationary distribution for any states s with $n_1 = \cdots = n_{d-1} = 0$ can now be expressed as

$$\pi(s) = \begin{cases} \bar{G}^i(K,d)\frac{\rho^i \pi(0)}{i!} & \text{for } i < d, \\ \bar{r}_i^{n_i} \ldots \bar{r}_d^{n_d} \bar{G}^i(K,d)\frac{\pi(0)}{i!}\rho^{\left(i+\sum_{j=d}^{i} n_j\right)} & \text{for } i \geq d. \end{cases} \tag{8}$$

Note that $i + \sum_{j=d}^{i} n_j$ represents the total number of jobs in the system. From (8), one directly concludes that the stationary distribution is a function only of ρ, K and d.

It is interesting to point out that the stationary distribution does not depend on the identity of the servers that are active since the servers are assumed to be homogeneous. Hence the stationary probabilities for states with the same number of active servers (i) and same number of waiting (n_i) jobs between servers are the same.

4.2. Normalization constant and probability of busy servers

Recall the notation that $p(i)$ denotes the stationary probability that the redundancy-d model with *c.o.s.* has i busy servers, for $1 \leq i \leq K$. This metric would be useful in obtaining the normalizing constant $\pi(0)$, which is also the probability that the system is empty ($p(0) = \pi(0)$). Noting that

$$p(i) = \sum_{s \in \mathcal{S}_i} \pi(s) \tag{9}$$

where $\mathcal{S}_i = \{s \in \mathcal{S} : \text{exactly } i \text{ servers are busy}\}$, we have the following lemma. The proof can be found in Appendix C.

Lemma 2. *The probability that i servers are busy is given by*

$$p(i) = \bar{p}(i)\pi(0) \tag{10}$$

where

$$\bar{p}(i) = \begin{cases} \binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i < d, \\ \left(\frac{1}{1-r_i}\right)\ldots\left(\frac{1}{1-r_d}\right)\binom{K}{i}\bar{G}^i(K,d)\rho^i, & \text{for } i \geq d. \end{cases} \tag{11}$$

Further, $\pi(0)$ is given by

$$\pi(0) = \left(1 + \sum_{i=1}^{K} \bar{p}(i)\right)^{-1}. \quad \square \tag{12}$$

A related performance metric of interest is $P_w(j)$, the probability that an arriving job sees j of its d feasible servers as busy. When a job arrives, conditioned on i busy servers, the probability that j of its servers are already busy is $\frac{\binom{K-j}{i-j}}{\binom{K}{i}}$, $i \geq j$. Therefore we have

$$P_w(j) = \sum_{i=j}^{K} p(i)\frac{\binom{K-j}{i-j}}{\binom{K}{i}}. \tag{13}$$

In particular, the probability a job has to wait, denoted by P_w, is given by $P_w(d)$.

4.3. Distribution of number of jobs

An important performance metric for the *c.o.s.* system is the number of waiting jobs. In this section, we obtain the expression for the probability-generating function for the number of waiting jobs in the system. An inversion of this transform gives us the distribution of the number of waiting jobs.

Define $p(i,m)$ as the probability that the system has i busy servers and m waiting jobs in the system. When $i < d$, there are no jobs waiting in the central queue and we have

$$p(i,m) = \begin{cases} p(i) & \text{for } m = 0, \\ 0 & \text{elsewhere.} \end{cases}$$

For $i \geq d$, it follows from Proposition 1 that

$$p(i,m) = \pi(0)\binom{K}{i}\bar{G}^i(K,d)\rho^i l_i(m)$$

$$\text{where } l_i(m) = \sum_{\substack{\{n_i \ldots n_d:\\ \sum_{l=d}^{i} n_l = m\}}} r_i^{n_i} \ldots r_d^{n_d}.$$

Let Q denote the random variable corresponding to the number of waiting jobs in the *c.o.s.* system. The probability that there are m waiting jobs ($Q = m$) is given by $\hat{p}(m) = \sum_{i=1}^{K} p(i, m)$. Using the above expressions for $p(i, m)$, we can derive the distribution for the number of waiting jobs. Its proof can be found in Appendix D.

Proposition 2. *The P.G.F. for the number of waiting jobs is given by*

$$E(z^Q) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} p(i) \left(\prod_{j=d}^{i} Geom_{r_j}(z) \right) \tag{14}$$

where

$$Geom_{r_j}(z) = \frac{1 - r_j}{1 - r_j z}$$

and $p(i)$ is given by Eqs. (10) and (12). The expected number of waiting jobs in the system is given by

$$E(Q) = \sum_{i=d}^{K} p(i) \left(\sum_{j=d}^{i} \frac{r_j}{1 - r_j} \right) \tag{15}$$

and the expected number of jobs in the system is given by $E(N) = E(Q) + \rho K$. □

Remark 1. From the P.G.F above, one can conclude that the random variable Q is a mixture of sum of geometric random variables.

Remark 2 (*Special Cases*)**.** We now discuss two special cases for the redundancy-d model with *c.o.s.* namely $d = 1$ and $d = K$. The case $d = 1$ or redundancy-1 *c.o.s.* corresponds to the system with Bernoulli routing of jobs to the K servers. This is also the case for redundancy-1 with *c.o.c*. The redundancy-K with *c.o.s.* is equivalent to an $M/M/K$ system. This is easy to see from the *central queue architecture*; since each job can be served at any of the K servers, a server that becomes free ends up choosing the head of the queue customer for service. We verify in Appendix E that our expressions for *c.o.s.* for these special cases indeed coincide with the known results.

5. Equivalence of JSW(d) and redundancy- d with *c.o.s.*

In this section, we will show the equivalence of the redundancy-d *c.o.s.* model with that of the join the shortest work among d servers (JSW(d)) policy. This is trivial to note for the case when $d = 1$ and $d = K$. When $d = 1$, the redundancy-1 *c.o.s.* model as well as the JSW(1) model is equivalent to probabilistic routing to K queues when the routing probabilities are Bernoulli ($\frac{1}{K}$). When $d = K$, redundancy-K *c.o.s.* model is equivalent to an $M/M/K$ multi-server queue (see Appendix E) whereas the *JSW*(K) policy is simply the traditional *JSW* (join shortest work) policy which is known to be equivalent to the $M/M/K$ queue [19]. While the equivalence is now apparent for $d = 1$ and $d = K$, we give a formal proof of this fact for any d in this section. The reasoning is based on a sample-path argument and hence the equivalence holds for generally distributed service requirements and heterogeneous servers.

Proposition 3. *Assume generally distributed service requirements and heterogeneous servers. For any given sample-path realization, a given job will be served under both JSW(d) and redundancy-d with c.o.s. in the same server. As a result, the following performance measures are the same under both models:*

- *Joint probability of servers being busy or idle.*
- *Delay distribution of a job.*
- *Total number of (waiting) jobs in the system.*

Proof. For a given realization, we couple the arrivals of jobs, the d servers sampled upon arrival, and their service requirements. For redundancy-d with c.o.s., a job cannot be overtaken by copies that arrive after him in any of its d sampled servers (since it has copies in all d FIFO-queues). It therefore follows directly that upon arrival of the job, one can determine the server in which the job will be served. In order to describe this server, we define the effective workload, $V_i^{red}(t)$ to denote the work present in server i that will be served in this server. This ignores the workload due to those copies in server i that will not be served in the server. As such, for redundancy-d with *c.o.s.* an arriving job will be served at the server with smallest workload $V_i^{red}(t)$, among the d servers.

Under JSW(d), an arriving job is sent to the server with smallest workload $W_i^{JSW}(t)$, among the d servers. Hence, in order to prove that a given job will be served in the same server under both models we are left to show that $W_i^{JSW}(t) = V_i^{red}(t)$ for all t. This can be seen as follows. First note that the (effective) workload decreases at rate 1 whenever the process is positive. Now assume $W_i^{JSW}(t) = V_i^{red}(t)$ is true at time t, and a new job arrives. (This is trivial to note for $t = 0$.) It is served in the server (among the d servers) with smallest value for $W_i^{JSW}(t)$ ($V_i^{red}(t)$), under JSW(d) (redundancy-d, respectively). Since

$W_i^{JSW}(t) = V_i^{red}(t)$, it is the same server say server m, that has the smallest value in the two systems and hence, the service requirement of this job is added both to $W_m^{JSW}(t^+)$ and to $V_m^{red}(t^+)$. □

Remark 3. With the above proposition, redundancy-d with *c.o.s.* can be perceived as a method of implementing JSW(d) without requiring the knowledge of the residual work in each of the queues. Such an implementation of JSW(d) would only be possible if the underlying system allows for the possibility to use extra copies of the same job. Indeed, the power of allowing even a single extra copy per job may prove to be very beneficial in the absence of workload information at the queues.

Now from the equivalence between JSW(d) and redundancy-d with *c.o.s.*, we have the following corollary that to the best of our knowledge provides performance metrics for JSW(d) that have not been obtained before.

Corollary 1. *For a JSW(d) system with Poisson arrivals and exponential service requirements, the P.G.F. for the number of waiting jobs is given by Eq. (14) and the expected number of waiting jobs in JSW(d) is given by Eq. (15). The expected number of jobs in JSW(d) is given by $E(N) = E(Q) + \rho K$.*

6. Asymptotic regimes for *c.o.s*

In this section, we consider the redundancy-d system with cancel-on-start under limiting regimes. We first consider the heavy-traffic regime where the number of servers is fixed and the traffic intensity approaches 1. In the second regime, we scale the number of servers and the arrival rate while keeping the traffic intensity unchanged. This is the mean-field regime.

6.1. Heavy-traffic regime

For the heavy-traffic analysis, we keep the number of servers, K, fixed and let $\frac{\lambda}{\mu} \uparrow K$, so that $\rho \uparrow 1$. When $d = 1$, we have a system of K independent $M/M/1$ queues with parameters λ/K and μ. Hence, after scaling by $1-\rho$, the total number of jobs in the system converges, as $\rho \uparrow 1$, to the sum of K exponentially distributed random variables with mean 1. In the result below, we derive the distribution of the total scaled number of waiting jobs for $d \geq 2$.

Proposition 4. *Assume $d \geq 2$. Then*

$$\lim_{\rho\uparrow 1} p(K) = 1 \quad \text{and} \quad \lim_{\rho\uparrow 1} E[e^{-s(1-\rho)Q}] = \frac{1}{1+s},$$

for any $s > 0$. In words, this implies that $(1-\rho)Q$ converges to an exponential random variable with unity parameter, as $\rho \uparrow 1$.

Proof. From Eq. (11) we have $\bar{p}(i) = O(\rho^i)$ for $i < d$, while $\bar{p}(K) = O(\rho^K/(1-\rho))$. Since $\rho \uparrow 1$, this implies that $\lim_{\rho\uparrow 1} p(K) = 1$. Now, substituting $z = e^{-s(1-\rho)}$ in Eq. (14), together with $\lim_{\rho\uparrow 1} p(K) = 1$ and taking the limit $\rho \to 1$ gives

$$\lim_{\rho\to 1} E[e^{-s(1-\rho)Q}] = \lim_{\rho\to 1} \left(\prod_{j=d}^{K} Geom_{r_j}(z) \right).$$

For $i < K$, we have $\lim_{\rho\to 1} r_i = \bar{r}_i < 1$, since $d \geq 2$. Further, $\lim_{\rho\to 1} r_K = \bar{r}_K = 1$. This implies that $\lim_{\rho\to 1} Geom_{r_j}(e^{-(1-\rho)s}) = 1$ for $j \neq K$. For the case $j = K$, using L'Hospital's rule it can be seen that $\lim_{\rho\to 1} Geom_{r_K}(e^{-(1-\rho)s}) = \frac{1}{1+s}$. This completes the proof. □

The above result shows that there is a large drop in the performance when passing from $d = 1$ to $d = 2$. Further, for $d \geq 2$, the performance is independent on the parameter d.

6.2. Mean-field regime

In this subsection we consider the mean-field regime and assume that the arrival rate into the system is $\lambda = \hat{\lambda}K$ where $\hat{\lambda} < \mu$, and let $K \to \infty$. We note that in a recent work Hellemans and van Houdt [18] have analyzed *JSW(d)* in the mean-field regime. In particular, they obtain the explicit expression for the limiting workload and response time distribution for exponential job sizes. In the following, we analyze the mean-field limit for the metric $P_w(j)$ due to its meaningfulness for *c.o.s.*

Define $\hat{\rho} = \frac{\hat{\lambda}}{\mu}$. For JSW($d$), asymptotic independence in the mean-field limit has been proved in [20]. This, together with the equivalence between JSW(d) and red-d *c.o.s.* and the fact that a given server under red-d *c.o.s.* is busy with probability $\hat{\rho}$ (see preliminaries), gives the following limiting result for the probability $P_w(j)$ that an arriving job sees j busy feasible servers.

Proposition 5. *For $\lambda = K\hat{\lambda}$ with $\hat{\lambda} < \mu$, we have*

$$\lim_{K\to\infty} P_w(j) = \hat{\rho}^j(1-\hat{\rho})^{d-j}\binom{d}{j}.$$

Proof. In [20, Theorem 2.1], JSW(d) is considered in the mean-field regime. It is proved there that given a finite set of servers, these servers behave independently in the mean-field limit. Note that the joint distribution of servers being idle is the same for both JSW(d) and red-d c.o.s. (Proposition 3). In addition, we showed that the probability a server is busy equals ρ. Hence, the probability that j servers are busy out of d, follows a binomial distribution with parameters d and $\hat{\rho}$. □

7. Redundancy with *c.o.c.*: an alternative product from

In this section, we give an alternative analysis for the redundancy-d model with *c.o.c.* [5,6] by generalizing the multi-type job and multi-type server model of Visschers et. al. [12]. Recall that the primary difference between redundancy-d *c.o.s.* and *c.o.c.* is as follows.

- In *c.o.c.*, redundant copies can be served simultaneously and are canceled once one of the copies receives service. When represented as a *central queue architecture*, it means that a job in the central queue can be served by multiple feasible servers simultaneously (see Fig. 1).
- The service rate of a job in the central queue for *c.o.c.* depends on the type of jobs that are ahead of it. Therefore, the departure rate of a job is in fact state dependent. (See [6] for more details.)

It is due to these two features of the *c.o.c.* model that a direct application of Theorem 1 is not possible.

In the rest of this section, we extend and adapt the model of Visschers et. al. [12] to accommodate these two features that are required for the *c.o.c.* model. In Section 7.1 we first present a generalization of [12] that allows for state dependent departure rates for servers. We call this the *generalized multi-type job and server model*. Then in Section 7.2, we adapt the state descriptor for the c.o.c. model so that it fits this *generalized multi-type job and server model*. These two steps allow us to provide an alternative product form stationary distribution for the *c.o.c.* model.

7.1. A generalized multi-type job and server model

Recall from the discussion in Section 3.3 that for the multi-type job and server model of [12], the *departure set function* $\mu_{\{\cdot\}}$ is defined by Eq. (3). In this section, we will show that it is possible to consider *departure set functions* that are different from Eq. (3) and that allow state dependent departure rates from active servers (without affecting the nature of the results in [12]). More specifically, we will model situations where in state $s = (n_i, M_i, \ldots, n_1, M_1)$, the service rate of any server M_j for $1 \le j \le i$ is a function of servers M_1 to M_{j-1} ahead of it. This is different from the setup of [12] where the service rate of server M_j was constant at all times. We begin by recalling the definition of $\hat{\mu}_{M_j}(s)$ from Section 3.3. For any state $s = (n_i, M_i, \ldots, n_1, M_1)$, $\hat{\mu}_{M_j}(s)$ is the departure rate from any server M_j with $1 \le j \le i$. We now have the following important definition.

Definition 1. A *generalized multi-type job and server model* is defined as a multi-type job and server model of [12] with the following modification.

1. The departure set function $\mu_{\{\cdot\}}$ is monotone, i.e., for every $\mathcal{A}_1, \mathcal{A}_2 \subseteq \mathcal{M}$ such that $\mathcal{A}_1 \subseteq \mathcal{A}_2$, we have $\mu_{\mathcal{A}_1} \le \mu_{\mathcal{A}_2}$.
2. In any state $s = (n_i, M_i, \ldots, n_1, M_1)$, we have

$$\hat{\mu}_{M_j}(s) := \mu_{\{M_1 \ldots M_j\}} - \mu_{\{M_1 \ldots M_{j-1}\}}. \tag{16}$$

From the above definition, note that the set function $\mu_{\{\cdot\}}$ can be defined in any arbitrary manner as long as it is monotone. The set function defined by Eq. (3) satisfies the two conditions above making the model of [12] a special case. In fact in that case we have $\hat{\mu}_{M_j}(s) = \mu_{\{M_j\}} = \mu_{M_j}$. Being a set function, it should be noted that $\mu_{\{M_1 \ldots M_i\}}$ also denotes total departure rate from any state $(n_i, M_{\sigma(i)}, \ldots, n_1, M_{\sigma(1)})$ where $\{\sigma(1), \ldots, \sigma(i)\}$ is a permutation of $\{1, \ldots, i\}$. In state $s = (n_i, M_i, \ldots, n_1, M_1)$, it goes without saying that the total departure rate $\mu_{\{M_1 \ldots M_i\}}$ satisfies

$$\mu_{\{M_1 \ldots M_i\}} = \sum_{j=0}^{i} \hat{\mu}_{M_j}(s)$$

(noting that for $j = 1$, we have $\mu_{\{M_1 \ldots M_{j-1}\}} = \mu_{\{\phi\}} = 0$). From the definition above, one can see that the departure rate from a *generalized multi-type job and server model* has the following features which are reminiscent of the *order independent* queue [15].

- The departure rate $\hat{\mu}_{M_j}(s)$ from an active server depends only on the active servers ahead of it.
- The total departure rate is independent of the permutation of active servers.

Before we proceed with the main result, we will provide an example of a *generalized multi-type job and server model* with three classes and three servers. The aim of this example is to present a scenario where servers have a state dependent departure rate.

Example 2. Let $\mathcal{C} = \{a, b, c\}$ and $\mathcal{M} = \{1, 2, 3\}$. Further, let $S_a = \{1\}$, $S_c = \{3\}$ and $S_b = \{1, 2, 3\}$. The *departure set function* for the model is defined as follows.

- $\mu_{\{1\}} = \mu_{\{2\}} = \mu_{\{3\}} = 1$
- $\mu_{\{1,2\}} = \mu_{\{2,3\}} = \mu_{\{1,3\}} = 1.5$ and
- $\mu_{\{1,2,3\}} = 1.5$.

For this model, consider a state of the form $s = (n_2, 2, n_1, 1)$. The total departure rate in this state is $\mu_{\{1,2\}} = 1.5$ and the departure rate from server 2 is given by $\hat{\mu}_2(s) = \mu_{\{2,1\}} - \mu_{\{1\}} = 0.5$. On the other hand, for some other state $s = (n_2, 1, n_1, 2)$, the departure rate from server 2 is equal to $\hat{\mu}_2(s) = \mu_{\{2\}} = 1$. Similarly, for state $s = (n_3, 2, n_2, 1, n_1, 3)$, we have $\hat{\mu}_2(s) = \mu_{\{2,1,3\}} - \mu_{\{1,3\}} = 0$. Clearly, this illustrates that the same server can have different service rates depending on the active servers ahead of it.

We now have the following proposition that upholds the validity of Theorem 1 for the *generalized multi-type job and server model*.

Proposition 6. *Consider a generalized multi-type job and server model with an assignment rule satisfying the assignment condition (Eq. (5)). Assuming the steady state exists, the steady state probability for any state $s = (n_i, M_i, \ldots, n_1, M_1) \in \mathcal{S}$ is given by*

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_1}^{n_1} \frac{\Pi_\lambda(\{M_1, \ldots, M_i\})}{\Pi_\mu(M_i, \ldots, M_1)} \pi(0) \tag{17}$$

where

$$\Pi_\lambda(\{M_1, \ldots, M_i\}) = \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots M_{j-1}\}),$$

$$\Pi_\mu(M_i, \ldots, M_1) = \prod_{j=1}^{i} \mu_{\{M_1, \ldots M_j\}},$$

$$\alpha_j = \frac{\lambda_{\mathcal{U}(\{M_1, \ldots, M_j\})}}{\mu_{\{M_1, \ldots M_j\}}}.$$

Proof. The proof of this proposition is similar to that of Theorem 2 of [12] (Theorem 1 in this paper). An outline of the proof is given in Appendix G to point out to the changes due to generalization of the *departure rate function*. □

7.2. An alternative product form for c.o.c.

In Proposition 6, we have generalized the results of [12] by considering a *generalized multi-type job and server model*. This generalization can now be used to apply Proposition 6 to redundancy-d with *c.o.c.*

For the redundancy-d *c.o.c.* model, one can view a type for a job as the choice of d servers that are chosen (see preliminaries). Since this choice is uniform over the set of all servers, each type is equally likely. Recall that λ_{type} denotes the arrival rate of each type where $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. Also recall some earlier notation where $\mathcal{C}$ denotes the set of all job types. For any type $c \in \mathcal{C}$, S_c denotes the set of d *feasible* servers for that type. Now for each type $c \in \mathcal{C}$, we associate a label O_c. This label will be used to identify the least recent (oldest) type c job present in the central queue.

A new state space representation for c.o.c.

We propose a new state space representation $(n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ to analyze the redundancy-d model of [6]. In such a state, the job at the head of line (of the central queue) is of type c_1. Since the central queue has FIFO ordering from right to left, this is also the oldest type c_1 job in state s and is therefore indicated by the label O_{c_1}. n_1 denotes the number of type-c_1 jobs that arrived after O_{c_1}. These jobs were followed by a type-c_2 job, represented by O_{c_2}. Now n_2 denotes the subsequent arrivals that are of either type c_1 or c_2. In general, in state s and for $0 \leq j \leq i$, O_{c_j} indicates the position in the central queue of the oldest type-c_j job and n_j denotes waiting jobs of type $c \in \bigcup_{i=1}^{j} c_i$. Now in state s, the jobs that can receive service are the ones that are represented by the labels (O_c). This is because of the FIFO scheduling in the central queue and the fact that the waiting jobs (n_j) are not the oldest jobs in their type. In state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, the departure rate of job O_{c_j} equals μ times the number of its feasible machines that are not used by the jobs $O_{c_{j-1}}, \ldots, O_{c_1}$. Hence, it is given by

$\hat{\mu}_{c_j}(s) := \mu\left(|F_j(s)| - |F_{j-1}(s)|\right)$, where $F_j(s) := \bigcup_{l=1}^{j} S_{c_l}$. The total departure rate in state s is the sum of the departure rates at each O_{c_j} for $0 \leq j \leq i$. This corresponds to the *departure set function* given by

$$\mu_{\{O_{c_1},\ldots O_{c_j}\}} = \sum_{t \in \bigcup_{l=1}^{j} S_{c_l}} \mu = \mu |F_j(s)|. \tag{18}$$

From the above description, it can be seen that the state space for the redundancy-2 *c.o.c.* model of Fig. 1 will be represented as $(O_{14}, O_{24}, O_{23}, O_{13}, O_{12})$. All the 5 jobs are of distinct classes and hence will be accorded distinct labels.

Redundancy-d c.o.c. as generalized multi-type job and server model

With a state space representation of $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, the *c.o.c.* model can now be viewed as a *generalized multi-type job and server model* where O_{c_j} resembles a super-server of state dependent service rate $\mu\left(|F_j(s)| - |F_{j-1}(s)|\right)$. Another artifact of this representation is that each super-server O_{c_j} can serve jobs only of type c_j and therefore each type has a dedicated super-server that will only serve jobs of its type. From this interpretation, it should be clear that there is no need to specify any *assignment rule* because an arriving job is never faced with the prospect of being served by multiple super-servers (although he will be served by multiple servers that define the super-server). Due to this, we have $\lambda_{O_c}(\{O_{c_1}, \ldots, O_{c_j}\}) = \lambda_c$ and hence the *assignment condition* is trivially satisfied. Proposition 6 can now be used to provide a new product form distribution for this redundancy model. This distribution is different from that of [6] due to a different state space representation. At the end of this section we comment on the latter. We have the following proposition (proof in Appendix H).

Proposition 7. *For the redundancy-d c.o.c. model, the steady state distribution for any generic state* $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ *is given by*

$$\pi(s) = \frac{\pi(0)}{i!} \prod_{j=1}^{i} \left(\frac{j\lambda_{type}}{\mu |F_j(s)|} \right)^{n_j+1}. \quad \square \tag{19}$$

The redundancy-*d c.o.c.* system has already been extensively studied by Gardner et al. [5,6] and hence we do not analyze this any further. The main purpose of Proposition 7 is to illustrate the unifying framework for redundancy models via Visschers et al. [12] and an adaptation of its generalization. To recall the main results from [6], the normalizing constant $\pi(0)$ in Proposition 7 is given by

$$\pi(0) = \prod_{i=d}^{K} \left(1 - \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}} \rho \right),$$

and the mean number of customers in the system $E(N)$ is

$$E[N] = \sum_{i=d}^{K} \frac{\rho}{\frac{\binom{K-1}{d-1}}{\binom{i-1}{d-1}} - \rho}. \tag{20}$$

Since our state space representation for analyzing *c.o.c.* is different from that used in [5,6], a direct comparison between Proposition 7 and [6, Theorem 2] is cumbersome. However one can compare certain states of the system to show that the stationary probabilities obtained from Proposition 7 and [6, Theorem 2] are indeed the same. For example, consider a state with m customers, all belonging to the same type. Such a state in our representation is denoted by $(m-1, O_c)$ and we have $i = 1$ and $F_1(s) = d$ in Proposition 7. This gives us

$$\pi((m-1, O_c)) = \pi(0) \left(\frac{\lambda_{type}}{d\mu} \right)^m.$$

This is the same as the steady state probability given in [6, Theorem 2]. Similarly, consider a state with m customers all from distinct types. In our representation, such a state is represented by $(O_{c_m}, \ldots, O_{c_1})$ and from Proposition 7,

$$\pi((O_{c_m}, \ldots, O_{c_1})) = \pi(0) \left(\frac{\lambda_{type}}{d\mu} \right)^m.$$

Again, this coincides with [6, Theorem 2].

8. Numerical results

In this section we present numerical results to further analyze the redundancy-*d* model considered in this paper. In Section 8.1 we numerically compare the performance between redundancy-*d* systems with *c.o.s.* and *c.o.c.* and study the impact of *independence assumption* made in the analysis of *c.o.c.* In Section 8.2, we provide some more numerical results for redundancy-*d c.o.s.* This is followed by numerical results for the mean-field regime in Section 8.3.

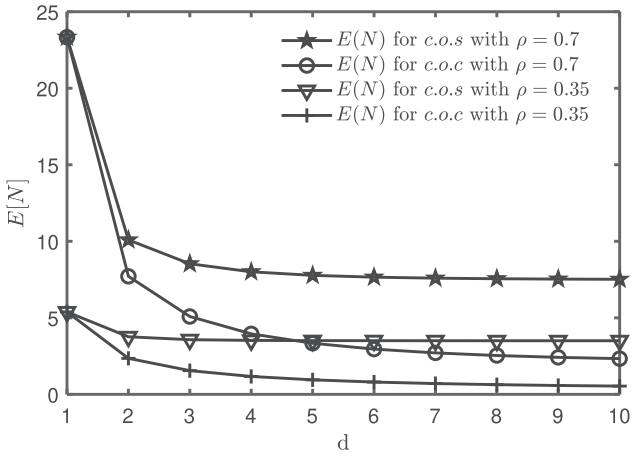

Fig. 2. $E(N)$ for *c.o.c.* and *c.o.s.* for $\rho = 0.35, 0.7$ and $K = 10$.

Table 1
Reduction in the mean number of jobs with *c.o.c.* and *c.o.s.* for $K = 10$, when increasing the redundancy parameter from $d = 1$ to $d = 2$.

ρ	$d = 1$ $E(N)$	% decrease in $E(N)$ for *c.o.s.* with $d = 2$	% decrease in $E(N)$ for *c.o.c.* with $d = 2$
0.1	1.11	9.5	51.56
0.2	2.5	18.17	53.3
0.3	4.28	26.25	55.26
0.4	6.66	33.94	57.5
0.5	10	41.44	60.09
0.6	15	48.95	63.17
0.7	23.33	56.75	66.95
0.8	40	65.28	71.79
0.9	90	75.44	78.56

8.1. Comparing *c.o.c.* and *c.o.s.*, impact of independence assumption

In this section we compare the performance of both systems and in particular assess the impact of the *independence assumption* (as made in *c.o.c.*) on the performance. Recall that due to cancellation on start, the analysis for *co.s.* with or without the independence assumption remains the same. We saw in Example 1 that while the *c.o.c.* model with the *independence assumption* significantly outperforms the c.o.s. model for $d = K$, the performance degrades severely when this assumption is dropped. In the rest of this section, we numerically analyze the impact of the *independence assumption* for arbitrary values of d.

We first compare the mean number of customers ($E(N)$) in the system under *c.o.s.* and *c.o.c.* The mean number of customers $E(N)$ in the *c.o.c.* system is given by Eq. (20), as derived in [6]. Consider *c.o.c.* and *c.o.s.* systems with parameters $K = 10$, $\rho = 0.35$ and 0.7. For varying values of d, Fig. 2 compares $E(N)$ for the two systems. The case $d = 1$ for both models is equivalent to Bernoulli routing to K servers in which case we have $E(N) = \frac{K\rho}{1-\rho}$. The case, $d = K$ for the *c.o.c.* model coincides with an $M/M/1$ server with arrival rate of λ and service rate of $K\mu$ and hence $E(N) = \frac{\rho}{1-\rho}$. On the other hand, $d = K$ for *c.o.s.* corresponds to an $M/M/K$ system with arrival rate λ and K servers each with a service rate 1. Naturally as expected, $E(N)$ for the $M/M/K$ system is larger than that of the $M/M/1$ system associated with $d = K$ for *c.o.c.* For $1 < d < K$, we see that $E(N)$ for the *c.o.c.* model is lower than that of the *c.o.s.* model and this is true for any value of ρ (illustrated in Fig. 2 for $\rho = 0.35$ and 0.7). What is also noticeable from the figure is that the proportion of reduction in $E(N)$ from having an extra copy ($d = 2$) in both the redundancy models depends on the load ρ in the system.

To elaborate on this, in Table 1, we compare $E(N)$ for *c.o.s.* and *c.o.c.* when increasing the redundancy parameter from $d = 1$ to $d = 2$. We fix $K = 10$ and $\mu = 1$ and take varying values of ρ. Note from the discussion after Proposition 1 that the stationary distribution for the *c.o.s.* system is merely a function of ρ, K and d. This is also the case for the *c.o.c.* system (see Proposition 7) and this justifies the choice of ρ as a parameter for this table. Table 1 indicates that $E(N)$ for *c.o.c.* system is lower than that of the *c.o.s.* system for various values of ρ. We also quantify the percentage reduction in $E(N)$ when increasing the redundancy parameter from $d = 1$ to $d = 2$.

A key observation from the table is the fact that for low values of ρ, the reduction with *c.o.s.* is smaller than that obtained with *c.o.c.* This is due to the independence assumption: at low loads, an arriving job will likely find the system empty. Under *c.o.c.*, the job will then get an instantaneous service rate of 2μ, which explains the reduction of the order of 50% in the table. However, with *c.o.s.*, even if the system is empty, an arriving job will only get served in one server. However, for higher

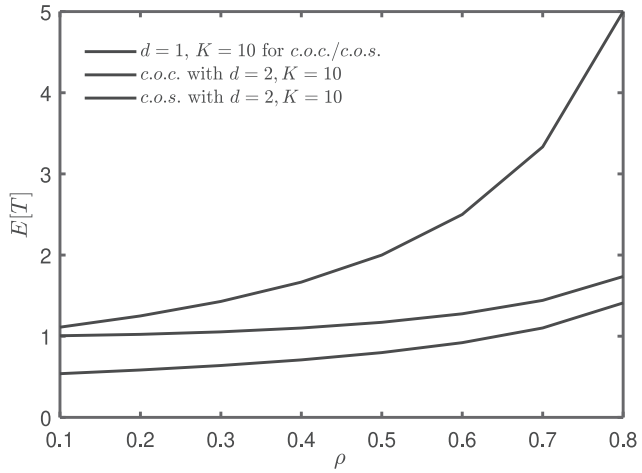

Fig. 3. $E(T)$ for $c.o.c.$ and $c.o.s.$ with $K = 10$.

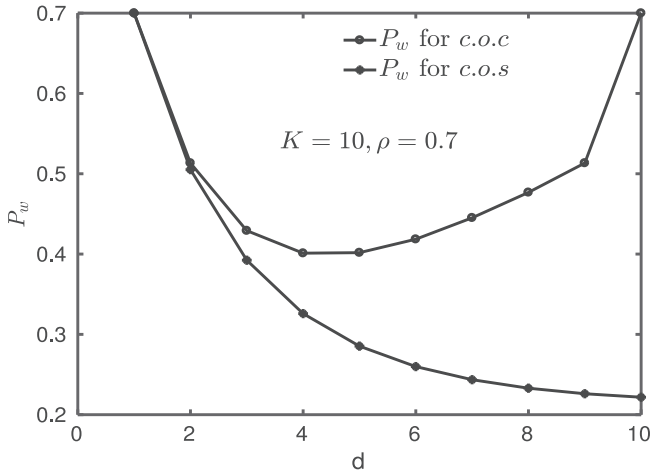

Fig. 4. P_w for $c.o.c.$ and $c.o.s.$ with $K = 10$, $\rho = 0.7$.

values of ρ, the gain obtained with $c.o.s.$ becomes comparable to that obtained with $c.o.c.$ In this case, the common feature that helps improve the performance with both systems is that by sending redundant copies, they can more efficiently use the capacity of the system.

In Fig. 3 we compare the mean sojourn time. We observe that almost up to $\rho = 0.7$, the mean sojourn time with $c.o.c.$ is below 1. Since the mean service time here is 1 ($\mu = 1$), this will not happen in most situations in practice. On the other hand, with $c.o.s.$, the mean sojourn time is always larger than 1. From the examples above, one is tempted to conclude the superiority of $c.o.c.$ over $c.o.s.$ This might seem counter-intuitive at first, since in the $c.o.c.$ model redundant copies can be served simultaneously, which implies extra work for the servers. The superiority of $c.o.c.$ system over $c.o.s.$ is primarily because of the independence assumption as made in $c.o.c.$ (see also the example in Section 3).

In Fig. 4 we plot the metric P_w, the probability that an arriving job has to wait with $K = 10$, $\rho = 0.7$ and for various values of d. We use Equation (5) from Gardner et al. [6] to obtain the numerical values for $p(i)$ in $c.o.c.$ The probability an arriving job has to wait under $c.o.c.$ can then be calculated using Eq. (13) with $j = d$. With $c.o.s.$ the probability that an arriving customer has to wait is lower than that for the $c.o.c.$ system. The probability to wait under $c.o.s.$ decreases with d, which is expected since as d increases, resources are more efficiently pooled together. With $c.o.c.$, for large values of d, the probability of waiting increases because multiple servers serve a single job and hence an arriving job is more likely to have its *feasible* servers to be busy. In spite of this, the performance of $c.o.c.$ is superior in terms of sojourn time (see Fig. 3), because as d increases, the system gets closer to being a single super-server (see also the discussion on $d = K$ in Section 3).

8.2. Performance of redundancy-d with c.o.s.

In Fig. 5 we plot the metric $p(i)$ (the probability that i servers are busy) for parameters $K = 10$, $\rho = 0.7$ and for different values of d. We observe from Fig. 5 that $p(K)$ increases in d. This is because a larger d implies a better utilization of all the servers and increases the probability of all servers being busy.

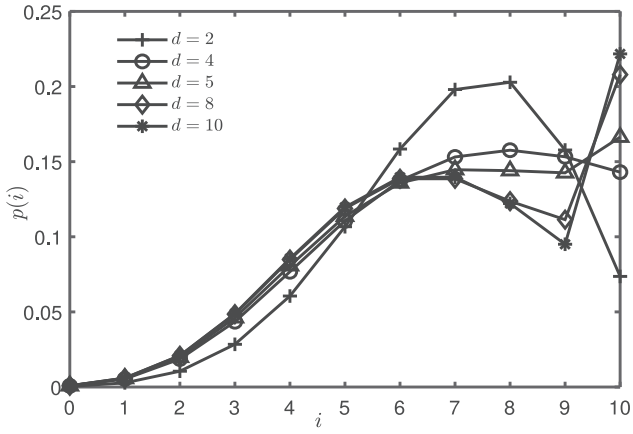

Fig. 5. $p(i)$ for different d when $K = 10$, $\rho = 0.7$.

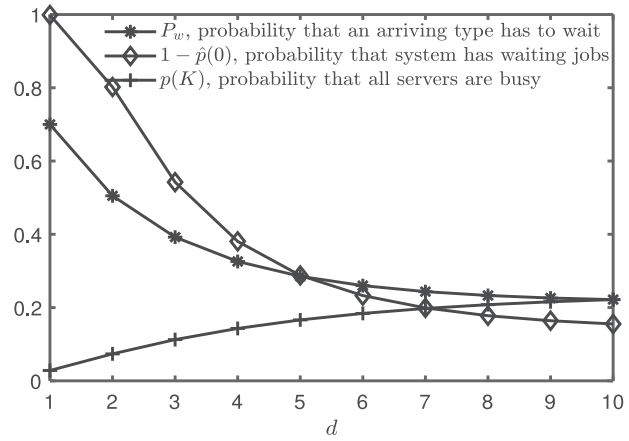

Fig. 6. P_w, $1 - \hat{p}(0)$ and $p(K)$ for $K = 10$ and $\rho = 0.7$.

This is however not the case in general for $p(i)$, for $2 \leq i < K$. In fact, $p(i)$ is increasing with d for lower values of i, (typically $i \leq 5$) and $p(i)$ is decreasing with d for higher values of i ($5 < i < 10$.) The effect reverses again for $i = K$ and we have $p(K)$ increasing in d. To explain this, let us consider the case $d = 10$. When $i < K$, $p(i)$ corresponds to having exactly i jobs present in the system. However, $p(10)$ is the probability that the number of jobs is equal or larger than 10, that is, it corresponds to the tail of the distribution. A similar reason holds for other values of d.

In Fig. 6, we examine the metrics P_w, $1 - \hat{p}(0)$ and $p(K)$ for the *c.o.s.* system with parameters $K = 10$ and $\rho = 0.7$ and for various values of d. While P_w denotes the probability that an arriving customer has to wait, $1 - \hat{p}(0)$ denotes the probability that the system has waiting customers. Note that $\hat{p}(0) = \sum_{i=0}^{d-1} p(i) + \sum_{l=d}^{K} p(l, 0)$. Fig. 6 illustrates the difference in the two quantities as a function of d. In fact, $P_w < 1 - \hat{p}(0)$ for $d < 5$, $P_w > 1 - \hat{p}(0)$ for $d > 5$ and $P_w = 1 - \hat{p}(0)$ for $d = 5$. From the figure, we also observe that for all d, $p(K) \leq P_w$ (with an equality when $d = K$). A justification for this is the fact that all servers being busy implies that an arriving customer has to wait. When $d = K$, an arriving customer having to wait implies that all servers are busy ($M/M/K$ system) and hence the equality.

8.3. Asymptotic regimes

In this section we focus on the mean-field regime. That is, we set $\lambda = \hat{\lambda}K$, for a fixed $\hat{\lambda}$, and let $K \to \infty$. In other words, the load per server is kept constant, while taking the number of servers to infinity. In Fig. 7 we plot P_w as a function of K and for different values of d. As expected, due to the asymptotic independence of the servers, P_w converges to $\hat{\rho}^d$ where $\hat{\rho} = \frac{\hat{\lambda}}{\mu}$ (see Proposition 5).

Recall that for $d = K$, the models *JSW*, redundancy-d *c.o.s.*, and $M/M/K$ are all the same. It is known for the $M/M/K$ system (and hence for *JSW*, see for example [21]) that in the mean field regime, the probability of waiting and the mean waiting time vanish in the limit. The same holds for the well-known *JSQ* dispatching policy. In a recent work, Mukherjee et al. [22] have shown that *JSQ*($d(K)$) with $d = o(K)$ can be asymptotically optimal in the sense that the probability of waiting and the mean waiting time are both equal to zero in the limit. The latter implies that *JSQ*($d(K)$) yields the same asymptotic performance as *JSQ*, but with a considerably smaller amount of overhead in terms of signaling (See the survey paper [23]

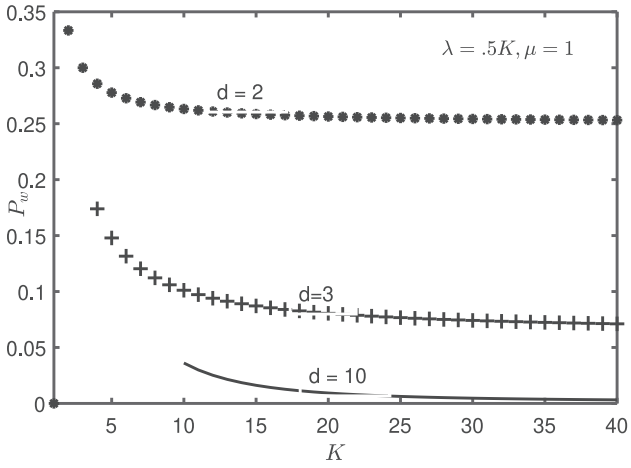

Fig. 7. $\lim_{K\to\infty} P_w = \hat{\rho}^d$ for $\hat{\lambda} = 0.5$ and fixed d.

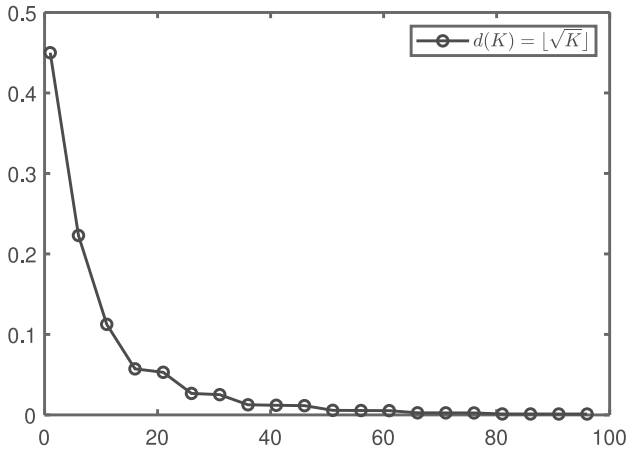

Fig. 8. Asymptotic optimality of $d(K) = \lfloor\sqrt{K}\rfloor$ for $\hat{\rho} = 0.45$.

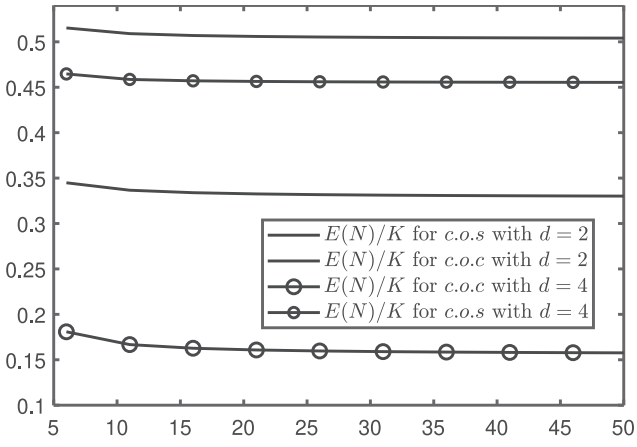

Fig. 9. Asymptotics for $E(N)/K$ for $c.o.s$ and $c.o.c$. system with $\hat{\lambda} = 0.45$ and various values of d..

for more details). Getting back to our model, an interesting question is then how big d needs to be, in order for $c.o.s.$ and $JSW(d)$, to be asymptotically optimal with respect to the waiting probability. In Fig. 8 we consider the case $d(K) = \lfloor\sqrt{K}\rfloor$, for $\hat{\rho} = 0.45$. This figure indicates that $c.o.s.$ might also be asymptotically optimal for an appropriate scaling of the redundancy parameter. A formal proof is left for future work.

Finally, we look at the asymptotic value of $E(N)/K$ for redundancy-d system with $c.o.s$ and $c.o.c$. This is illustrated in Fig. 9. We can derive a simple approximation for the mean sojourn time under $c.o.s.$ and hence for the mean number of jobs in the system. If an incoming job does not need to wait, its sojourn time will be $1/\mu$. If it has to wait, then its sojourn time will be

its waiting time plus its service time $1/\mu$. To calculate the waiting time in the latter case, we assume that there are no other jobs (with feasible servers in common) waiting in front of the job. Hence, the waiting time will be equal to the minimum of d exponential service times, i.e. $1/(\mu d)$. Now, recalling that the probability of waiting in the limit is $\hat{\rho}^d$ (Proposition 5), and using Little's law, the mean number of jobs for K sufficiently large can be approximated by:

$$E(N) \approx \hat{\lambda}K\left((1-P_w)\frac{1}{\mu}+P_w(\frac{1}{\mu d}+\frac{1}{\mu})\right)=\hat{\rho}K\left(1+\frac{\hat{\rho}^d}{d}\right).$$

For the values of Fig. 9, for $d=2$ the approximation yields 0.496 whereas the value from the plots is 0.504, and for $d=4$ these values are 0.4646 and 0.4553, respectively. For sufficiently large K, we expect the approximation to get more accurate as the load decreases and as the parameter d increases, since then the probability of waiting decreases. Indeed, as $K\to\infty$ in the mean-field regime, the above approximation coincides with the first order approximation of the corresponding mean-field result for $JSW(d)$ given by Theorem 5.4 in [18].

9. Concluding remarks

We have obtained a unifying framework to investigate the steady state distribution of $c.o.s.$ and $c.o.c.$ redundancy models. In the literature there are already several studies for the $c.o.c.$ model, but to the best of our knowledge, we are the first to give an exact analysis of $c.o.s.$ for any $1\le d\le K$. In addition, due to the equivalence between $c.o.s.$ and $JSW(d)$, our approach provides a first exact analysis of workload-based load balancing schemes.

There are several interesting research problems that stem from our work. An important extension is to investigate $c.o.s.$ with heterogeneous servers and class-based models. Our approach based on the multi-type multi-server queue will still be valid, however, the routing discipline that will satisfy the assignment condition, see Eq. (5), needs to be determined. In this paper we have mostly focused on the queue length and related performance metrics. Obviously, the waiting time and the sojourn time distribution are very relevant metrics as well. A good approach would be to extend the central queue approach to study the waiting time distribution of both $c.o.s.$ and $c.o.c.$ models. Another research path pertains with a thorough analysis of c.o.s. model in the mean-field regime. The proof for asymptotic optimality of JSW($d(K)$) is part of future work. Last but not least, the analysis of $c.o.c.$ without the independence assumption is a very relevant problem, and it would be of interest to study whether this model can be analyzed through a similar approach as the one developed in this paper.

Acknowledgment

The authors would like to thank Dr. Ayalvadi Ganesh from the University of Bristol for constructive comments that helped improve the presentation of the paper.

Appendix A. Proof of Lemma 1

Assuming the *uniform assignment rule*, we will prove that for every subset $\{M_i,\dots,M_1\}$ of $\mathcal{M}$ of size i and for every j $(1\le j\le i)$ we have $\lambda_{M_j}(\{M_1,\dots,M_{j-1}\})=\lambda_{\bar{M}_j}(\{\bar{M}_1,\dots\bar{M}_{j-1}\})$ for every permutation $\bar{M}_1,\dots\bar{M}_i$ of $M_1,\dots,M_i$. This will imply the *assignment condition*.

To characterize $\lambda_{M_j}(\{M_1,\dots,M_{j-1}\})$ (the activation rate of server M_j, when the set of active servers is $\{M_1,\dots,M_{j-1}\}$), we will require the number of types that can activate server M_j when servers M_1 to M_{j-1} are busy. This can be obtained by first counting the number of types that can activate server M_j while having exactly α of its d *feasible* servers already busy (and hence among M_1 to M_{j-1}) and then summing over α.

Now let $C(m_i)$ denote the set of types which have m_i as a *feasible* server. Likewise, $C(M_i)$ would denote the same for a generic server M_i. Recall that S_c denotes the set of d *feasible* servers for type c jobs. Now when the system has exactly $j-1$ busy servers denoted by $\{M_1,\dots,M_{j-1}\}$ and when M_j is an idle server, define

$$C^{\alpha}_{M_j}(\{M_1,\dots,M_{j-1}\}) := \left\{c\in C(M_j):|S_c\cap\{M_1,\dots,M_{j-1}\}|=\alpha\right\}$$

as the set of types that have the idle server M_j as *feasible* and have exactly α out of their d *feasible* servers in the set $\{M_1,\dots,M_{j-1}\}$. Hence for any type $c\in C^{\alpha}_{M_j}(\{M_1,\dots,M_{j-1}\})$, there are exactly $d-\alpha$ idle servers (which include server M_j) and due to the *uniform assignment rule*, an arriving customer of this type will activate server M_j with probability $\frac{1}{d-\alpha}$. Now it can be seen that

$$|C^{\alpha}_{M_j}(\{M_1,\dots,M_{j-1}\})|=\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}.$$

Here $\binom{j-1}{\alpha}$ is the number of ways of choosing α servers out of $\{M_1,\dots,M_{j-1}\}$. For a particular choice of α servers say $\{M^1,\dots,M^{\alpha}\}$, and given M_j, the number of types that have these particular $\alpha+1$ servers as *feasible* servers is $\binom{K-j}{d-\alpha-1}$. This explains the value of $|C^{\alpha}_{M_j}(\{M_1,\dots,M_{j-1}\})|$. Note additionally that this cardinality depends only on the number of busy

servers $j-1$ and not on the label of the servers that are actually busy. This implies that for any permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots, M_i$ and for $0 \le j \le i$, we have

$$|C^{\alpha}_{M_j}(\{M_1, \ldots, M_{j-1}\})| = |C^{\alpha}_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})|. \tag{21}$$

Here $\{\bar{M}_1, \ldots \bar{M}_{j-1}\}$ is the first j servers from the permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots, M_i$ for $0 \le j \le i$. Now note that

$$\begin{aligned}
&\lambda_{M_j}(\{M_1, \ldots, M_{j-1}\}) \\
&= \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \sum_{c \in C^{\alpha}_{M_j}(\{M_1,\ldots,M_{j-1}\})} \frac{\lambda_{type}}{d-\alpha} \\
&= \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\lambda_{type}|C^{\alpha}_{M_j}(\{M_1, \ldots, M_{j-1}\})|}{d-\alpha} \\
&= \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\lambda_{type}|C^{\alpha}_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})|}{d-\alpha} \\
&= \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\})
\end{aligned}$$

where $\{\bar{M}_1, \ldots \bar{M}_{j-1}\}$ is the first j servers from the permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots, M_i$ for $0 \le j \le i$. Here the range of α in the index of the summation is so chosen that the quantity $\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}$ is property defined. The first equality above follows from the fact the each type $c \in C^{\alpha}_{M_j}(\{M_1, \ldots, M_{j-1}\})$ activates its idle feasible server M_j with rate $\frac{\lambda_{type}}{d-\alpha}$ and the total activation rate for M_j is obtained by summing over all such types and then over all α. The third equality follows from Eq. (21) and the last equality in fact follows from the first. From this, it is easy to see that for $i = 1 \ldots K$,

$$\prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots, M_{j-1}\}) = \prod_{j=1}^{i} \lambda_{\bar{M}_j}(\{\bar{M}_1, \ldots \bar{M}_{j-1}\}) \tag{22}$$

for every permutation $\bar{M}_1, \ldots \bar{M}_i$ of $M_1, \ldots, M_i$. This completes the proof. □

Appendix B. Proof of Proposition 1

For a redundancy-d system with *c.o.s*, first recall that if less than d servers are busy, then the number of waiting copies (in the traditional parallel server representation) is zero. This is because a waiting copy implies that all the other $d-1$ copies are also waiting for service; this is possible when at least d servers are busy in the system. Therefore for any *feasible* state $s = (n_i, M_i, \ldots, n_1, M_1)$ we have $n_1 = n_2 = \cdots - n_{d-1} = 0$. Since the redundancy-$d$ system with *c.o.s* can be seen as a multi-type job and server model and since the *uniform assignment rule* satisfies the *assignment condition* (Lemma 1), we can apply Theorem 1. The steady state distribution for any state $s = (n_i, M_i, \ldots, n_1, M_1)$ with $i < d$ and $n^1 = \cdots = n^{d-1} = 0$ is now given by

$$\pi(s) = \frac{\Pi_{\lambda}(\{M_1, \ldots, M_i\})}{\Pi_{\mu}(M_i, \ldots, M_1)}\pi(0)$$

and for $i \ge d$ and $n^1 = \cdots = n^{d-1} = 0$ by

$$\pi(s) = {\alpha_i}^{n_i} \ldots {\alpha_d}^{n_d} \frac{\Pi_{\lambda}(\{M_1, \ldots, M_i\})}{\Pi_{\mu}(M_i, \ldots, M_1)}\pi(0).$$

Here

$$\begin{aligned}
\Pi_{\lambda}(\{M_1, \ldots, M_i\}) &= \prod_{j=1}^{i} \lambda_{M_j}(\{M_1, \ldots, M_{j-1}\}) \\
\Pi_{\mu}(M_i, \ldots, M_1) &= \prod_{j=1}^{i} \mu_{\{M_1,\ldots,M_j\}} \\
\alpha_j &= \frac{\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}}{\mu_{\{M_1,\ldots,M_j\}}} \text{ for } 1 \le j \le i.
\end{aligned}$$

For the redundancy-d system with *c.o.s*, the departure rate from any state with active servers $\{M_1, \ldots, M_j\}$ is given by $\mu\{M_1, \ldots, M_j\} = j\mu$ and hence

$$\Pi_{\mu}(M_i, \ldots, M_1) = \mu^i i!.$$

Recall from the preliminaries that $\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})}$ denotes the arrival rate of types that can be served only at servers $\{M_1,\ldots,M_j\}$. The number of such types is $\binom{j}{d}$ and the arrival rate of each such type is $\lambda_{type} = \frac{\lambda}{\binom{K}{d}}$. Therefore

$$\lambda_{\mathcal{U}(\{M_1,\ldots,M_j\})} = \lambda \frac{\binom{j}{d}}{\binom{K}{d}}.$$

Now using the fact that

$$\begin{aligned}
&\lambda_{M_j}(\{M_1,\ldots,M_{j-1}\}) \\
&= \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{|C^{\alpha}_{M_j}(\{M_1,\ldots,M_{j-1}\})|}{d-\alpha} \\
&= \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha},
\end{aligned}$$

we have

$$\begin{aligned}
\Pi_{\lambda}(\{M_1,\ldots,M_i\}) &= \prod_{j=1}^{i} \lambda_{M_j}(\{M_1,\ldots,M_{j-1}\}) \\
&= \prod_{j=1}^{i} \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha}.
\end{aligned}$$

Now define

$$\begin{aligned}
G_j(K,d) &:= \lambda_{type} \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{d-\alpha} \\
&= \sum_{\alpha=max(0,j-1+d-K)}^{min(j-1,d-1)} \frac{\lambda\binom{j-1}{\alpha}\binom{K-j}{d-\alpha-1}}{(d-\alpha)\binom{K}{d}} \\
G^i(K,d) &:= \prod_{j=1}^{i} G_j(K,d) \text{ and} \\
r_i &:= \left(\lambda \frac{\binom{i}{d}}{i\mu\binom{K}{d}}\right)
\end{aligned}$$

Using these definitions, the statement of the theorem follows. This completes the proof. □

Appendix C. Proof of Lemma 2

When $i < d$ it is easy to see that

$$p(i) = \binom{K}{i} i!\pi(s)$$

where s is any state of the form $s = (0, M_i, 0, M_{i-1}, \ldots, 0, M_1)$ (since the servers are homogeneous, $\pi(s)$ for all such states is the same). The i busy servers can be chosen in $\binom{K}{i}$ ways and there are $i!$ ways to arrange the servers leading to the above expression.

Recall that $r_i = \rho\bar{r}_i$. Since $\rho < 1$, and $\bar{r}_i = \frac{\binom{i-1}{d-1}}{\binom{K-1}{d-1}} < 1$, it directly follows that $r_i < 1$. The probability of having i busy servers, $p(i)$, for $i \geq d$ is now given by

$$\begin{aligned}
p(i) &= \binom{K}{i} i! \sum_{n_i} \cdots \sum_{n_d} \pi(n^i, M_i, \ldots, n^d, M_d, \ldots, 0, M_1) \\
&= \binom{K}{i} i! \sum_{n_i} \cdots \sum_{n_d} {r_i}^{n_i} \ldots {r_d}^{n_d} G^i(K,d) \frac{\pi(0)}{i!\mu} \\
&= \binom{K}{i} \left(\frac{1}{1-r_i}\right) \cdots \left(\frac{1}{1-r_d}\right) G^i(K,d) \frac{\pi(0)}{\mu^i}.
\end{aligned}$$

Therefore we have

$$p(i) = \begin{cases} \binom{K}{i} G^i(K,d) \frac{\pi(0)}{\mu^i} & \text{for } i < d \\ \binom{K}{i} \left(\frac{1}{1-r_i}\right) \dots \left(\frac{1}{1-r_d}\right) G^i(K,d) \frac{\pi(0)}{\mu^i} & \text{for } i \geq d. \end{cases}$$

Recall that $\binom{K}{i} \frac{G^i(K,d)}{\mu^i} = \binom{K}{i} \bar{G}^i(K,d) \rho^i$ and hence $\bar{p}(i)$ given in the statement now follows. Using the fact that $\sum_{i=0}^{K} p(i) = 1$ and $p(i) = \bar{p}(i)\pi(0)$, one can see that

$$\pi(0) = \left(1 + \sum_{i=1}^{K} \bar{p}(i)\right)^{-1}. \quad \square$$

Appendix D. Proof of Proposition 2

We first analyze $l_i(m)$ and try to obtain a recursive form for it. First note that for $i = d$, we have $l_d(m) = r_d^m$ for $m \geq 0$. Now for $i > d$,

$$\begin{aligned} l_i(m) &= \sum_{\substack{\{n_i \dots n_d: \\ \sum_{l=d}^{i} n_l = m\}}} r_i^{n_i} \dots r_d^{n_d} \\ &= \sum_{n_i=0}^{\infty} r_i^{n_i} \sum_{\substack{\{n_{i-1} \dots n_d: \\ \sum_{l=d}^{i-1} n_l = m - n_i\}}} r_{i-1}^{n_{i-1}} \dots r_d^{n_d} \\ &= \sum_{n_i=0}^{m} r_i^{n_i} l_{i-1}(m - n_i) \\ &= (h_i \star l_{i-1})(m) \end{aligned}$$

where $h_i(j) := r_i^j$ and $\star$ represents the convolution operator. We will now obtain the expression for $P_i(z)$, the P.G.F. for $p(i, m)$. Let $H_i(z)$ and $L_i(z)$ be the P.G.F.'s for $h_i(\cdot)$ and $l_i(\cdot)$ respectively. When there are $i < d$ servers busy, there are no jobs waiting. Hence, we directly have $P_i(z) = p(i)$, for $i < d$.

Let

$$f^i(K,d) = \binom{K}{i} \bar{G}^i(K,d) \rho^i.$$

For $i \geq d$, we have $p(i, m) = \pi(0) f^i(K, d) l_i(m)$. Further note when $i \geq d$, we have

$$\begin{aligned} H_i(z) &= \frac{1}{1 - r_i z} \\ L_{i+1}(z) &= H_{i+1}(z) L_i(z) \text{ and} \\ P_i(z) &= \pi(0) f^i(K,d) L_i(z), \end{aligned}$$

and since $l_d(m) = r_d^m$, we have $L_d(z) = H_d(z)$. We therefore have for $i \geq d$,

$$P_i(z) = \pi(0) f^i(K,d) \left(\prod_{j=d}^{i} H_j(z)\right), \tag{23}$$

where $\pi(0)$ is given by Eq. (12). Now define

$$Geom_{r_j}(z) := \frac{1 - r_j}{1 - r_j z}.$$

$Geom_{r_j}(z)$ denotes the P.G.F. of a *geometric* random variable X_j with parameter r_j. (X_j has the distribution $P(X_j = n) = (1 - r_j) r_j^n$.). Together with Lemma 2, we then have for $i \geq d$

$$\begin{aligned} P_i(z) &= \pi(0) f^i(K,d) \left(\prod_{j=d}^{i} H_j(z)\right) \\ &= p(i) \left(\prod_{j=d}^{i} Geom_{r_j}(z)\right). \end{aligned} \tag{24}$$

The P.G.F. for the number of waiting jobs is then given by

$$\hat{P}(z) = \sum_{i=1}^{K} P_i(z) = \sum_{i=0}^{d-1} p(i) + \sum_{i=d}^{K} P_i(z).$$

The required P.G.F. in the statement of the theorem now follows from Eq. (24).

The expected number of waiting jobs is given by $\sum_{i=1}^{K} \frac{dP_i(z)}{dz}\Big|_{z=1}$. Note that $\frac{dP_i(z)}{dz} = 0$ when $i < d$. When $i \geq d$,

$$\begin{aligned}\frac{dP_i(z)}{dz}\Big|_{z=1} &= p(i)\left(\prod_{j=d}^{i} Geom_{r_j}(z)\right)\Big|_{z=1}\left(\sum_{j=d}^{i}\frac{r_j}{1-r_j}\right)\\ &= p(i)\left(\sum_{j=d}^{i}\frac{r_j}{1-r_j}\right).\end{aligned}$$

(The second equality is because $\left(\prod_{j=d}^{i} Geom_{r_j}(z)\right)\Big|_{z=1} = 1$.) The expected number of waiting jobs in the system ($E(Q)$) is hence given by

$$E(Q) = \sum_{i=d}^{K} p(i)\left(\sum_{j=d}^{i}\frac{r_j}{1-r_j}\right)$$

where $p(i)$ is given by Eq. (10) and (12). Since the mean number of jobs at the servers is same as the mean number of busy servers we have $E(N) = E(Q) + K\rho$. □

Appendix E. Special cases; $d = 1$ and $d = K$

In this appendix, we will discuss the special cases for the *c.o.s.* model corresponding to $d = 1$ and $d = K$.

When $d = 1$, both *c.o.s.* and *c.o.c.* systems are equivalent to a system with Bernoulli routing to the K servers. The queue length process of each server is independent and each server represents an $M/M/1$ queue with arrival rate $\frac{\lambda}{K}$ and service rate μ. It is easy to check from our analysis of Section 4 that when $d = 1$, we have $G_j(K,d) = \frac{\lambda}{K}$, $G^i(K,d) = \left(\frac{\lambda}{K}\right)^i$ and $r_i = \rho$. Using the notation that $\rho = \frac{\lambda}{K\mu}$ (the load per $M/M/1$ queue), the steady-state probability from Eq. (7) for any state $s = (n_i, M_i, \ldots, n_1, M_1)$ for $i \geq 0$ is given by

$$\pi(s) = \pi(0)\frac{\rho^{\left(i+\sum_{j=1}^{i} n_j\right)}}{i!}.$$

$p(i)$ is given by $\bar{p}(i)\pi(0)$ where from Eq. (11) we have $\bar{p}(i) = \binom{K}{i}\left(\frac{\rho}{1-\rho}\right)^i$. From Eq. (12), the normalizing constant is obtained as

$$\begin{aligned}\pi(0) &= \left(1 + \sum_{i=1}^{K}\binom{K}{i}\left(\frac{\rho}{1-\rho}\right)^i\right)^{-1}\\ &= \left(\left(1+\frac{\rho}{1-\rho}\right)^K\right)^{-1} = (1-\rho)^K.\end{aligned}$$

Since $\pi(0)$ is also the probability that the system of K independent $M/M/1$ queues is empty, this justifies $\pi(0) = (1-\rho)^K$. Using this, $p(i)$ can be simplified to $p(i) = \binom{K}{i}\rho^i(1-\rho)^{K-i}$. Now using Proposition 2, it is easy to see that the P.G.F. for the number of waiting jobs in the redundancy-1 system is given by

$$\hat{P}(z) = \left(\frac{1+\rho-\rho z}{1-\rho z}\right)^K (1-\rho)^K.$$

Further, the mean number of waiting jobs and the total number of jobs in the system are given by $E(Q) = \frac{K\rho^2}{1-\rho}$ and $E(N) = K\frac{\rho}{1-\rho}$.

We now analyze the **redundancy-K** system with *c.o.s.* The redundancy-K system is same as an $M/M/K$ system with arrival rate of λ and K servers each with rate μ. For this case, substituting $d = K$ in our analysis of Section 4 gives us $G_j(K,K) = \frac{\lambda}{K-j+1}$, $G^i(K,K) = \frac{\lambda^i(K-i)!}{K!}$ and $r_K = \rho$. Again, by simple substitution, the probability of i busy servers $p(i)$ is given by $p(i) = \pi(0)\bar{p}(i)$ where

$$\bar{p}(i) = \begin{cases}\frac{(K\rho)^i}{i!} & \text{for } i < K\\ \left(\frac{1}{1-\rho}\right)\frac{(K\rho)^K}{K!} & \text{for } i = K.\end{cases}$$

The normalizing constant $\pi(0)$ is given by

$$\pi(0) = \left(\sum_{i=0}^{K-1} \frac{(K\rho)^i}{i!} + \left(\frac{1}{1-\rho} \right) \frac{(K\rho)^K}{K!} \right)^{-1}$$

which coincides with the normalizing constant of an $M/M/K$ system (see page 260, [19]). Note that $p(K)$, the probability that all servers are busy corresponds to the blocking probability of the $M/M/K$ system. In fact, we obtain

$$p(K) = \pi(0) \left(\frac{1}{1-\rho} \right) \frac{(K\rho)^K}{K!}$$

which as expected coincides with the celebrated Erlang-C formula. Based on substituting $d = K$ in Proposition 2, we see that the P.G.F. for the number of waiting jobs in the redundancy-K system is given by

$$\hat{P}(z) = \sum_{i=0}^{K-1} p(i) + p(K) Geom_\rho(z).$$

Further, the mean number of waiting jobs and total number of jobs in the system are given by $E(Q) = p(K)\frac{\rho}{1-\rho}$ and $E(N) = p(K)\frac{\rho}{1-\rho} + \frac{\lambda}{\mu}$. These expressions verify with the corresponding quantities for the $M/M/K$ system (see [19]).

Appendix F. Transition rates and global balance

In this appendix, we will outline the state transitions and the global balance equations for the *multi-type job and server* model of [12]. In Section 7.1, we consider a *generalized multi-type job and server model* that differs from the model of [12] only in its *departure set function*. The state transitions and the global balance equations described below hold for this generalized model as well.

The transitions from an arbitrary state are of the following type.

Arrivals: When in state $s = (n_i, M_i, \ldots, n_1, M_1)$, the total arrival rate is λ. An arriving job either finds no *feasible* server to be idle or is picked for service by a *feasible* idle server. In case of multiple *feasible* idle server, the *assignment rule* will determine which of the *feasible* and idle server will serve this arrival. The arrival rate of jobs that find no *feasible* server to be idle is given by $\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}$. The arrival rate of jobs that activate a *feasible* server (according to the *assignment rule*) is $\lambda - \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})}$ respectively. We have

$$\lambda - \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} = \sum_{M \in \mathcal{M} \setminus \{M_1,\ldots,M_i\}} \lambda_M(\{M_1, \ldots, M_i\}). \tag{25}$$

Departures: On departure of a job in service, either the server serving this job becomes idle or the server serving this job picks some waiting job. These two cases are defined by the following two events that describe transition rates associated with a departure leading into state s.

- { Rate P }: the average rate of transition into state s due to a departure such that the server serving the departing job becomes idle.
- { Rate Q }: the average rate of transition into state s due to a departure such that the server serving the departing job picks a waiting job.

The global balance equations for any state $s = (n_i, M_i, \ldots, n_1, M_1)$ are as follows. When $n_i > 0$, we have

$$\begin{aligned} \pi(s)\left(\lambda + \mu_{\{M_1,\ldots M_i\}}\right) &= \{ \text{Rate P} \} + \{ \text{Rate Q} \} \\ &+ \lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} \pi(n_i - 1, M_i, \ldots, n_1, M_1). \end{aligned} \tag{26}$$

When $n_i = 0$, we have

$$\begin{aligned} \pi(s)\left(\lambda + \mu_{\{M_1,\ldots M_i\}}\right) &= \{ \text{Rate P} \} + \{ \text{Rate Q} \} \\ &+ \lambda_{M_i}(\{M_1, \ldots, M_{i-1}\}) \pi(n_{i-1}, M_{i-1}, \ldots, n_1, M_1). \end{aligned} \tag{27}$$

We will now characterize Rates P and Q in detail. As in [12], we denote the state $(n_i, M_i, \ldots, l, M, n_k - l, M_k, \ldots, n_1, M_1)$ by $insert_{kl}^{M}(s)$ and the state

$$\begin{aligned} &(n_i, M_i, \ldots, n_{j+1}, M_{j+1}, n_j + n_{j-1} + 1, M_{j-1}, \ldots, \\ &\ldots, M_{k+1}, l, M_j, n_k - l, M_k, \ldots, n_1, M_1) \end{aligned}$$

by $swap_{kl}^{M_j}(s)$. Note that in state $insert_{kl}^{M}(s)$ (resp. $swap_{kl}^{M_j}(s)$), the departure rate of server M (resp. M_j) in our system is given by

$$\mu_{\{M_1,\ldots,M_k,M\}} - \mu_{\{M_1,\ldots,M_k\}} (\text{resp.} \mu_{\{M_1,\ldots,M_k,M_j\}} - \mu_{\{M_1,\ldots,M_k\}}).$$

Define

$$\delta_j(M) := \frac{\lambda_{\mathcal{U}(\{M_1 \ldots M_j\})}}{\lambda_{\mathcal{U}(\{M_1 \ldots M_j, M\})}}, \quad j = 1 \ldots i.$$

Let $p_{kl}^{M}(s)$ (resp. $q_{kl}^{M_j}(s)$) denote the probability of transition from state $insert_{kl}^{M}(s)$ (resp. $swap_{kl}^{M_j}(s)$) to state s due to departure of server M (resp. M_j). Therefore $p_{kl}^{M}(s)$ is the probability that after departure of server M, the state $insert_{kl}^{M}(s)$ is such that there are no waiting jobs in the system that are *feasible* to server M. Similarly, $q_{kl}^{M_j}(s)$ is the probability that after departure of server M_j, the state $swap_{kl}^{M_j}(s)$ is such that after departure of server M_j, the next *feasible* job for this server is the $n_{j-1}+1^{st}$ job after server M_{j-1}. Therefore we have,

$$p_{kl}^{M}(s) = \delta_k(M)^l \delta_{k+1}(M)^{n_{k+1}} \dots \delta_i(M)^{n_i}$$

and

$$q_{kl}^{M_j}(s) = \delta_k(M_j)^l \delta_{k+1}(M_j)^{n_{k+1}} \dots \delta_{j-1}(M_j)^{n_{j-1}}(1-\delta_{j-1}(M_j)).$$

The rates P and Q can be quantified as below:

$$\{\text{ Rate P }\} = \sum_{M\in\mathcal{M}\setminus\{M_1,\dots,M_i\}} P_M(s)$$

$$\{\text{ Rate Q }\} = \sum_{j=1}^{i} Q_{M_j}(s)$$

where

$$\begin{aligned} P_M(s) &= \sum_{k=1}^{i}\sum_{l=0}^{n_k} \hat{\mu}_M(insert_{kl}^{M}(s)) p_{kl}^{M}(s)\pi(insert_{kl}^{M}(s)) \\ &\quad + \hat{\mu}_M(s,0,M) p_{00}^{M}(s)\pi(s,0,M), \\ Q_{M_j}(s) &= \sum_{k=1}^{j-1}\sum_{l=0}^{n_k} \hat{\mu}_{M_j}(swap_{kl}^{M_j}(s)) q_{kl}^{M_j}(s)\pi(swap_{kl}^{M_j}(s)) \\ &\quad + \hat{\mu}_{M_j}(swap_{00}^{M_j}(s)) q_{00}^{M_j}(s)\pi(swap_{00}^{M_j}(s)). \end{aligned}$$

Here, for the model of [12], we have $\hat{\mu}_M(insert_{kl}^{M}(s)) = \mu_M$ and $\hat{\mu}_{M_j}(swap_{kl}^{M_j}(s)) = \mu_{M_j}$. For the *generalized model* of Section 7.1, $\hat{\mu}_{\cdot}(\cdot)$ is given by Eq. (16).

Appendix G. Proof outline of Proposition 6

The proof outline is as follows. We first note that the state transitions and the global balance equation of Appendix F also hold for the *generalized multi-type job and server model* with a departure set function $\mu_{\{\}}$. Note the subtle difference between our definition of $P_M(s)$, $Q_{M_j}(s)$ given in the appendix above and that in [12]. Unlike [12], we have subsumed the departure rates inside the definition of $P_M(s)$, $Q_{M_j}(s)$ to be able to consider more general *departure set functions* for the generalization. Now as in [12] we shall identify the partial balance equations that need to be satisfied by our guess for the stationary distribution given in the statement of the theorem. Using the conditions in the definition of the *generalized model*, we show that the stationary distribution from the theorem indeed satisfies the partial balance resulting in a product-form.

Partial Balance: Now as in the proof of Theorem 2, [12], the aim is to prove that the expression for $\pi(s)$ in the statement of the theorem satisfies the following 4 partial balance equation. For $n_i > 0$

$$\pi(s)\mu_{\{M_1,\dots M_i\}} = \lambda_{\mathcal{U}(\{M_1,\dots,M_i\})}\pi(n_i-1, M_i, \dots, n_1, M_1) \tag{28}$$

and for $n_i = 0$,

$$\begin{aligned} &\pi(s)\mu_{\{M_1,\dots M_i\}} = \\ &\lambda_{M_i}(\{M_1,\dots,M_{i-i}\})\pi(n_{i-1}, M_{i-1}, \dots, n_1, M_1). \end{aligned} \tag{29}$$

$$\begin{aligned} \pi(s)\lambda_{\mathcal{U}(\{M_1,\dots,M_i\})} &= \{\text{ Rate Q }\} \\ &= \sum_{j=1}^{i} Q_{M_j}(s) \end{aligned} \tag{30}$$

$$\pi(s)\lambda_M(\{M_1,\dots,M_i\}) = P_M(s). \tag{31}$$

Here we have made use of Eq. (25) to replace λ in the global balance equations (26) and (27). Firstly, it is easy to see that Eq. (6) satisfies Eqs. (28) and (29). Hence to complete the proof, it is sufficient to prove that Eq. (6) satisfies Eqs. (30) and (31). Let us first prove that Eq. (6) satisfies Eq. (31). We proceed exactly as in the proof of Theorem 2 in [12] and divide each term in the summation of $P_M(s)$ by $\pi(s)\lambda_M(\{M_1,\dots,M_i\})$. In the corresponding part of the proof of Theorem 2 in [12], we have μ_M

instead of $\hat{\mu}_M(insert_{kl}^M(s))$. Proceeding in the same way and using the fact that $\hat{\mu}_M(insert_{kl}^M(s)) = \mu_{\{M_1,\ldots,M_k,M\}} - \mu_{\{M_1,\ldots,M_k\}}$ it can be shown that

$$\begin{aligned}&\hat{\mu}_M(insert_{kl}^M(s))p_{kl}^M(s)\frac{\pi(insert_{kl}^M(s))}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})}\\ &= (1-\beta_k)(\beta_k)^l(\beta_{k+1})^{n_{k+1}+1}\ldots(\beta_i)^{n_i+1}\end{aligned}$$

where

$$\beta_j = \frac{\mu_{\{M_1,\ldots,M_j\}}}{\mu_{\{M_1,\ldots,M_j,M\}}} 1 \le j \le i.$$

A similar procedure for the boundary state gives us

$$\hat{\mu}_M(s,0,M)p_{1n_1}^M(s)\frac{\pi(s,0,M)}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})} = \prod_{j=1}^{i}(\beta_j)^{n_j+1}.$$

Now as in [12], this implies that

$$\begin{aligned}&\frac{P_M(s)}{\pi(s)\lambda_M(\{M_1,\ldots,M_i\})} =\\ &\left[\prod_{j=1}^{i}(\beta_j)^{n_j+1} + \sum_{k=1}^{i}\sum_{l=0}^{n_k}(1-\beta_k)(\beta_k)^l\prod_{h=k+1}^{i}(\beta_h)^{n_h+1}\right]\\ &= 1.\end{aligned}$$

To prove, that Eq. (6) satisfies Eq. (30), as in the proof of Theorem 2 in [12], divide each term in the summation of $Q_{M_j}(s)$ by $\pi(s)$. Since the departure rate $\hat{\mu}_{M_j}(swap_{kl}^{M_j}(s))$ in state $swap_{kl}^{M_j}(s)$ is given by $\mu_{\{M_1,\ldots,M_k,M_j\}} - \mu_{\{M_1,\ldots,M_k\}}$, we can follow the exact steps in the proof of Theorem 2 in [12] to obtain

$$\begin{aligned}&\hat{\mu}_{M_j}(swap_{kl}^{M_j}(s))q_{kl}^{M_j}(s)\frac{\pi(swap_{kl}^{M_j}(s))}{\pi(s)}\\ &= \left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right)\\ &\times (1-\beta_{k,j})(\beta_{k,j})^l(\beta_{k+1,j})^{n_{k+1}+1}\ldots(\beta_{j-1,j})^{n_{j-1}+1}\end{aligned}$$

where

$$\beta_{h,j} = \frac{\mu_{\{M_1,\ldots,M_h\}}}{\mu_{\{M_1,\ldots,M_h,M_j\}}}, 1 \le h \le j-1.$$

For the boundary states, we can show

$$\begin{aligned}&\hat{\mu}_{M_j}(swap_{00}^{M_j}(s))q_{1,n_1}^{M_j}(s)\frac{\pi(swap_{00}^{M_j}(s))}{\pi(s)} =\\ &\left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right)\prod_{h=1}^{j-1}(\beta_{h,j})^{n_h+1}\end{aligned}$$

and that for $1 \le j \le i$, adding terms over k and l we would get

$$Q_{M_j}(s) = \pi(s)\left(\lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1},M_j\})} - \lambda_{\mathcal{U}(\{M_1,\ldots,M_{j-1}\})}\right).$$

Summing over all j we have

$$\pi(s)\lambda_{\mathcal{U}(\{M_1,\ldots,M_i\})} = \sum_{j=1}^{i} Q_{M_j}(s)$$

which is same as Eq. (30). This completes the proof outline. □

Appendix H. Proof of Proposition 7

From Section 3, recall that the steady state exists for any $\rho < 1$. Since the redundancy-d *c.o.c* model can be viewed as a *generalized multi-type job and server model* with state space representation $(n_i, O_{c_i}, \ldots, n_1, O_{c_1})$, we are in position to invoke Proposition 6 for this model. Therefore the steady state distribution in state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ is given by

$$\pi(s) = {\alpha_i}^{n_i}\ldots{\alpha_1}^{n_1}\frac{\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\})}{\Pi_\mu(O_{c_i},\ldots,O_{c_1})}\pi(0) \tag{32}$$

where

$$\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\}) = \prod_{j=1}^{i} \lambda_{O_{c_j}}(\{O_{c_1},\ldots O_{c_{j-1}}\}),$$

$$\Pi_\mu(O_{c_i},\ldots,O_{c_1}) = \prod_{j=1}^{i} \mu_{\{O_{c_1},\ldots O_{c_j}\}},$$

$$\alpha_j = \frac{\lambda_{\mathcal{U}(\{O_{c_1},\ldots,O_{c_j}\})}}{\mu_{\{O_{c_1},\ldots O_{c_j}\}}}.$$

Now for the redundancy-d $c.o.c$ model, we have $\lambda_{O_{c_j}}(\{O_{c_1},\ldots O_{c_{j-1}}\}) = \lambda_{c_j} = \lambda_{type}$. Further $\lambda_{\mathcal{U}(\{O_{c_1},\ldots,O_{c_j}\})} = \sum_{l=1}^{j} \lambda_{c_l} = j\lambda_{type}$. Therefore we have,

$$\Pi_\lambda(\{O_{c_1},\ldots,O_{c_i}\}) = \prod_{j=1}^{i} \lambda_{type},$$

$$\alpha_j = \left(\frac{j\lambda_{type}}{\mu|F_j(s)|}\right).$$

After rearranging the terms and bit of algebra, the steady state distribution for state $s = (n_i, O_{c_i}, \ldots, n_1, O_{c_1})$ is given by

$$\pi(s) = \frac{\pi(0)}{i!} \prod_{j=1}^{i} \left(\frac{j\lambda_{type}}{\mu|F_j(s)|}\right)^{n_j+1} \quad \square \tag{33}$$

References

[1] G. Ananthanarayanan, S. Kandula, A.G. Greenberg, I. Stoica, Y. Lu, B. Saha, E. Harris, Reining in the outliers in map-reduce clusters using mantri, in: OSDI, vol. 10, 2010, p. 24.

[2] A. Vulimiri, P.B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, S. Shenker, Low.latency.via. redundancy, Low latency via redundancy, in: Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies, ACM, 2013, pp. 283–294.

[3] G. Joshi, E. Soljanin, G. Wornell, Queues with redundancy: Latency-cost analysis, ACM SIGMETRICS Perf. Eval. Rev. 43 (2) (2015) 54–56.

[4] N.B. Shah, K. Lee, K. Ramchandran, When do redundant requests reduce latency? IEEE Trans. Commun. 64 (2) (2016) 715–722.

[5] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytiä, A. Scheller-Wolf, Queueing with redundant requests: exact analysis, Queueing Syst. 83 (3–4) (2016) 227–259.

[6] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, S. Zbarsky, Redundancy-d: The power of d choices for redundancy, Operations Research.

[7] G. Ananthanarayanan, A. Ghodsi, S. Shenker, I. Stoica, Effective straggler mitigation: Attack of the clones, in: NSDI, vol. 13, 2013, pp. 185–198.

[8] J. Dean, L.A. Barroso, The tail at scale, Commun. ACM 56 (2) (2013) 74–80.

[9] K. Lee, N.B. Shah, L. Huang, K. Ramchandran, The mds queue: Analysing the latency performance of erasure codes, IEEE Trans. Inform. Theory 63 (5) (2017) 2822–2842.

[10] K. Lee, R. Pedarsani, K. Ramchandran, On scheduling redundant requests with cancellation overheads, IEEE/ACM Trans. Netw. 25 (2) (2017) 1279–1290.

[11] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, B. Van Houdt, A better model for job redundancy: Decoupling server slowdown and job size, IEEE/ACM Trans. Netw. 25 (6) (2017) 3353–3367.

[12] J. Visschers, I. Adan, G. Weiss, A product form solution to a system with multi-type jobs and multi-type servers, Queueing Syst. 70 (3) (2012) 269–298.

[13] T. Bonald, C. Comte, Balanced fair resource sharing in computer clusters, Perform. Eval. 116 (2017) 70–83.

[14] T. Bonald, C. Comte, F. Mathieu, Performance of balanced fairness in resource pools: A recursive approach, Proc. ACM Meas. Anal. Comput. Syst. 1 (2) (2017) 41.

[15] A. Krzesinski, Order.independent. queues, Order independent queues Queueing Networks (2011) 85–120.

[16] I. Adan, R. Righter, G. Weiss, FCFS parallel service systems and matching models, in: Valuetools, 2017.

[17] G. Joshi, E. Soljanin, G. Wornell, Efficient redundancy techniques for latency reduction in cloud systems, ACM Trans. Model. Perform. Eval. Comput. Syst. (TOMPECS) 2 (2) (2017) 12.

[18] T. Hellemans, B. Van Houdt, On the power-of-d-choices with least loaded server selection, Proc. ACM Meas. Anal. Comput. Syst. 2 (2) (2018) 27.

[19] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, New York, NY, USA, 2013.

[20] M. Bramson, Y. Lu, B. Prabhakar, Asymptotic independence of queues under randomized load balancing, Queueing Syst. 71 (3) (2012) 247–292.

[21] S. Halfin, W. Whitt, Heavy-traffic limits for queues with many exponential servers, Oper. Res. 29 (3) (1981) 567–588.

[22] D. Mukherjee, S.C. Borst, J.S. van Leeuwaarden, P.A. Whiting, Universality of power-of-d load balancing in many-server systems, arXiv preprint arXiv:1612.00723.

[23] M. van der Boor, S.C. Borst, J.S. van Leeuwaarden, D. Mukherjee, Scalable load balancing in networked systems: A survey of recent advances, arXiv preprint arXiv:1806.05444.

Urtzi Ayesta received a PhD degree from Université de Nice-Sophia Antipolis (France), a MS degree in Electrical Engineering from Columbia University (US) and a BS/MS degree in Telecomunication Engineering from Nafarroako Unibertsitate Publikoa-Universidad Publica de Navarra (Spain). His PhD research work was carried out at the research laboratories of INRIA Sophia-Antipolis and France Telecom R&D. He is currently a CNRS researcher working at IRIT, Toulouse, and he also holds an adjunct lecturer position (part-time appointment funded by Ikerbasque) in the Computer Science Faculty at the University of the Basque Country.

Tejas Bodas received his M. Tech and Ph.D degree in 2016 from the Department of Electrical Engineering at the Indian Institute of Technology in Mumbai (IIT Bombay). He was a postdoc at LAAS, CNRS in Toulouse, France from 2016 to 2018 and is currently a (short term) postdoc at the Department of Mathematics and Computer Science in the University of Antwerp, Belgium.

Ina Maria (Maaike) Verloop received the M.Sc. degree in Mathematics from Utrecht University, The Netherlands, in 2005, and the Ph.D. degree from the Mathematics and Computer Science Department of Eindhoven University of Technology, in 2009. Her Ph.D. research was carried at the Probability, Networks and Algorithms Department of the Center for Mathematics and Computer Science (CWI) in Amsterdam, The Netherlands. From 2009 to 2011 she was a postdoc at the Basque Center for Applied Mathematics, Derio, Spain. Since October 2011 she has been a CNRS researcher at IRIT, Toulouse, France.